TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

OLLI RITAKALLIO

AUTOMAATION SOVELLUSSUUNNITTELUN LAADUNVARMISTUKSEN PROSESSIN KEHITTÄMINEN

Diplomityö

Tarkastaja: Prof. Hannu Koivisto
Tarkastaja ja aihe hyväksytty
Teknisten tieteiden tiedekuntaneuvoston kokouksessa 4. maaliskuuta 2015

TIIVISTELMÄ

OLLI RITAKALLIO: Automaation sovellussuunnittelun laadunvarmistuksen prosessin kehittäminen
Tampereen teknillinen yliopisto
Diplomityö, 102 sivua, 7 liitesivua
huhtikuu 2016
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Automaation tietotekniikka
Tarkastaja: Prof. Hannu Koivisto
Avainsanat: laadunvarmistus, automaatio, sovellussuunnittelu

Tämän työn myötä kehitetään Insta Automation Oy:n automaation sovellussuunnittelun laadunvarmistusprosessia. Kehityksellä pyritään vastaamaan automaatiosovellusten laadullisiin haasteisiin. Sovellussuunnittelussa esiintyvät päävaiheet tukitoimintoineen sekä laadunvarmistukseen käytettävät menetelmät tunnistettiin kirjallisuudesta. Näiden avulla muodostettiin kehitysehdotus organisaation sovellussuunnittelun laadusta varmistumiseksi.

Kehitysehdotuksen muodostaminen pohjautui siihen olettamaan, että laatu muodostuu sovellussuunnittelussa systemaattisen suunnitteluprosessin mukaisella kurinalaisella ohjelmointityöllä sekä projektinaikaisilla tukitoimilla. Laadunvarmistuksen tukitoiminnolla yritys vaikuttaa tuotteensa laadun riittävyyteen. Laadunvarmistus rakentuu todentamisen ja kelpoistamisen toiminnoista. Todentamisessa on kyse toteutettavan sovelluksen testaamisesta sekä sovelluksen dokumentaation että koodin verifioinnista. Menetelminä tähän käytetään katselmoinnin ja tarkastusten lisäksi epämuodollisia tilaisuuksia. Suunnittelun aikana tuotettu laadunvarmistuksen dokumentaatio toimii laadun todistusaineistona, jota asiakas käyttää toimitetun järjestelmän kelpoistamisessa.

Laadunvarmistuksen kehittämiseksi ensin dokumentoitiin yrityksen automaation sovellussuunnittelun prosessi asiantuntijahaastatteluiden avulla. Sovellussuunnittelun työprosessista muodostettiin organisaation toimintajärjestelmään geneerinen malli, joka soveltuu kaikille toimialasegmenteille. Ohjeellinen prosessimalli toimii suunnannäyttäjänä ja viitekehyksenä sovellussuunnittelutyössä sekä sovelluksen laadunvarmistamisessa, jolla tässä tarkoitetaan kaikkia niitä työtehtäviä jotka suoritetaan vaatimusmäärittelystä aina lopullisen järjestelmän hyväksyntään saakka.

ABSTRACT

OLLI RITAKALLIO: The development of quality assurance process of automation application design
Tampere University of Technology
Diplomityö, 102 pages, 7 Appendix pages
April 2016
Master's Degree Programme in Automation Engineering
Major: Automation information technology
Examiner: Prof. Hannu Koivisto
Keywords: Quality assurance, automation, software

The development of quality assurance of automation application desig in Insta Automation Oy is the main subject of this thesis. The QA development adresses the challenges of automation application design. During the development the main charachters of software design were identified from the literature to further develop the process and an initiative was formed.

The initiative is based on the assumption that quality of software is build by a systematic design process supported by QA activities. These activities are intended to help reaching the appropriate level of quality. The QA itself is based on verification & validation. In the framework of software design the V&V provides means to test the application and to inspect documents related to it. Formal and informal methods are used. The formal methods provides documented proofs of the quality which are intended for the customer to validate the commissioned industrial system.

Design process of automation application was firstly documented. This was conducted by interviewing the company experts. Based on the documented process a model of the appropriate design was supplied to companys quality management system. The formulated model includes the documented process which is reorganized to fit better with future QA activites. In the suggested new process model the emphasis is on the software design description, system testing and tecnical review. Also the objectivity of V&V is something not to forget. The formulated model is intended to quide the design which includes all the work design engineer faces in the context of automation application design and to assure its quality.

ALKUSANAT

Vuoden 2015 kesäajan käytin tämän diplomityön kirjallisuusaineiston kokoamiseen. Kahlasin aineistoa läpi samalla kun vietin aikaani rannalla yksivuotiaan poikani kanssa. Tuo aika oli hyvin antoisaa myös senkin takia, että aikaisemmin täysin tuntematon aihe muodostui mielenkiintoiseksi.

Erityiset kiitokset kuuluu aiheen minulle antaneelle Jarkko Järvilehdolle. Hän myös mahdollisti työn sujuvan etenemisen diplomityöpalaverien avulla, joissa sain suuntaa antavaa palautetta työn sisällöstä. Kiitokset kuuluvat kaikille asiantuntijahaastatteluihin osallistuneille, jotka löysivät sitä varten aikaa projektikiireiden lomassa. Sekä kiitokset myös Joonas Eirolalle, joka antoi työn asiasisällöstä asiantuntijaarvion.

Työn aihealue muodostui hyvin laajaksi. Mitä enemmän softasuunnittelun laadunvarmistukseen perehtyi, sitä enemmän löytyi uusia aihesuuntauksia. Tämän myötä työn laajuus lähti paisumaan. Kirjallisuusosuuden (review) toikin päätökseen tilanne, jossa uudet lähteet toistivat samoja jo aikaisemmin havaittuja seikkoja. Tämän tilanteen jälkeen lähteistä kootut muistiinpanot muodostivat perustan, jonka mukaan varsinainen työn oma osuus (case) on rakennettu. Aihealueen rajaamiseen toikin suurta apua Hannu Koiviston diplomityön ohjaustuokiot. Nämä tarjosivat oivaa apua yhtenäisemmän, johdonmukaisemman ja tiiviimmän diplomityön muodostamiseksi.

Kaikkein suurimmat ja erityisimmät kiitokset kuuluvat omalle pienelle perheelleni, joka omalta osaltaan mahdollisti tämän työn maaliin viemisen.

Tampereella, 8.5.2016

Olli Ritakallio

SISÄLLYS

1. Johdanto . . . 1
1.1 Työn aihealue ja tavoitteet . . . 1
1.2 Tutkimusmenetelmät . . . 3
1.3 Työn rakenne . . . 4
2. Sovellussuunnitteluprosessi . . . 6
2.1 Prosessiajattelu . . . 6
2.2 Prosessin kehittäminen . . . 7
2.3 Sovellussuunnittelun vaiheet . . . 8
2.3.1 Määrittelyvaihe . . . 9
2.3.2 Suunnitteluvaihe . . . 10
2.3.3 Toteutusvaihe . . . 11
2.3.4 Käyttöönotto . . . 12
2.4 Automaatiosovelluksen suunnittelu . . . 13
2.5 Turvallisuuskriittinen automaatio . . . 15
2.5.1 Lääketeollisuus . . . 17
2.5.2 Turvallisuuteen liittyvät järjestelmät . . . 17
2.5.3 Ydinvoimateollisuus . . . 18
3. Elinkaarimalleja . . . 19
3.1 Automaation elinkaari . . . 19
3.2 Vesiputous- ja v-malli . . . 20
3.3 Inkrementaaliset elinkaarimallit . . . 23
3.4 Ketterät menetelmät . . . 25
4. Laadunhallinnan periaatteet . . . 27
4.1 Laadunhallintajärjestelmä . . . 27
4.2 Laadun mittaaminen . . . 29
4.3 Toteutettavan tuotteen laadunhallinta . . . 30
4.4 Laadunvarmistuksen ongelmat . . . 31
4.5 Katsaus aikaisemmista lähestymistavoista . . . 32
4.6 CMMI kypsyysmallin mukainen lähestymistapa . . . 34
4.7 Automaation laatutekijät . . . 35
4.8 Ohjelmistojen laatuterminologiaa . . . 36
4.8.1 Laatupoikkeamat . . . 37
4.8.2 Laatumetriikka . . . 39
5. Projektin tukitoiminnot . . . 40
5.1 Laadunvarmistus . . . 40
5.1.1 Todentaminen ja kelpoistaminen . . . 41
5.1.2 Läpikäynti . . . 44
5.1.3 Tekninen katselmointi . . . 46
5.1.4 Testaaminen . . . 47
5.1.5 Tarkastus . . . 49
5.1.6 Tarkastuksen eroavaisuudet tekniseen katselmointiin . . . 52
5.2 Vaatimustenkäsittely . . . 53
5.2.1 Vaatimusmäärittely . . . 54
5.2.2 Vaatimustenhallinta . . . 55
5.2.3 Vaatimusten jäljitettävyys . . . 56
5.3 Konfiguraationhallinta . . . 57
5.3.1 Versionhallinta . . . 58
5.3.2 Muutostenhallinta . . . 60
5.4 Dokumentointi . . . 62
6. Dokumentoitu sovellussuunnitteluprosessi . . . 64
6.1 Prosessin kuvaaminen . . . 64
6.2 Haastatteluaineiston mukainen sovellussuunnitteluprosessi . . . 65
6.2.1 Sovellussuunnittelu . . . 65
6.2.2 Käyttöönotto . . . 71
6.3 Organisaation segmenttikohtaiset suunnitteluprosessin vaiheet . . . 72
6.3.1 Elintarvike- ja kemianteollisuus . . . 73
6.3.2 Koneistot ja näyttämöt . . . 74
6.3.3 Ympäristö- ja vesihuolto . . . 75

6.3.4 Energia- ja prosessiteollisuus 76
6.4 Suunnitteluprosessista tunnistetut elinkaarimallien yhtäläisyydet . . . 77
7. Laadunvarmistusprosessin kehittäminen 79
7.1 Ohjeellinen sovellussuunnittelun prosessimalli 80
7.1.1 Laadunvarmistusprosessi osana ohjeellista prosessimallia 82
7.1.2 Ohjelmistokuvauksen merkitys 86
7.1.3 Laadunvarmistuksen mekanismeilla tuotettavia laadun todisteita 88
7.2 Ohjeellisen prosessimallin soveltaminen eräässä automaatioprojektissa 90
7.2.1 Laadunvarmistussuunnitelma 91
7.2.2 Asiantuntija-arvio kehitysehdotuksen soveltuvuudesta 93
7.3 Laadunvarmistuksen prosessin kehittämisehdotus 94
7.3.1 Lyhyen aikavälin kehitys 95
7.3.2 Pitkän aikavälin kehitys 96
8. Yhteenveto ja päätelmät 97
Lähteet 99

KUVALUETTELO

1.1 Diplomityön rakenne 5

3.1 Automaation elinkaari 21
3.2 Vesiputousmalli 22
3.3 V-malli 23

4.1 Taulukko laatuominaisuuksista 38

5.1 Tarkastustilaisuuden eteneminen 50
5.2 Vaatimustenkäsittely 54
5.3 Vaatimustenhallinnan CMMI:n mukainen esitys 56
5.4 Sovelluksen versionumerointi 59
5.5 Muutostenhallinnan yleinen esitys 61
5.6 Taulukko automaatiosovelluksen dokumentaatiosta 62

6.1 Dokumentoitu sovellussuunnittelu 66

7.1 Ohjeellisen automaation sovellussuunnitteluprosessin päävaiheet. . . . 80
7.2 Ohjeellisen suunnitteluvaiheen pääsisältö, menodokumentaatio ja tulokset. 81
7.3 Ohjeellinen toteutusvaihe. 83
7.4 Käyttöönoton vaiheet. 83
7.5 Ohjelmistokuvauksen merkitys. 88
7.6 Taulukko laadun todisteista 89

TERMIT JA LYHENTEET

big bang	ohjelmiston integroiminen kertaoperaationa
black box	testaamisessa ohjelmiston toiminnallisuuden testaaminen
bottom-up	yksityskohdista aloittettu kokonaisuuteen eteneminen
CMMI	Combatibility Maturity Model Integration, kypsyystason arviointi- ja kehitysmalli
CPU	Central Processing Unit, ohjelmoitavan logiikan keskusyksikkö
DCS	Distributed Control System, hajautettu automaatiojärjestelmä
ERP	Enterprice Resource Planning, toiminnanohjausjärjestelmä
ESW	Embedded Software, reaaliaikajärjestelmän sulautettu ohjelmisto
faceplate	operointipaneeli
FAT	Factory Acceptance Test, tehdastestaus
firmware	sulautettu ohjelma tai laiteohjelma
GAMP	Good Automated Manufacturing Practices, ohjeistus lääketeollisuuden automaatiototeutuksille
glass box	testaamisessa ohjelmiston sisäisen rakenteen testaaminen
GMP	Good Manufacturin Practices, lääketeollisuuden hyvät tuotantotavat
Harmony	reaaliaikajärjestelmän ketterä kehitysprosessi
I/O	Input/Output, tulo/lähtö liityntä
IAEA	International Atomic Energy Agency, kansainvälinen atomienergiajärjestö
ISA-95	kansainvälinen standardi ohjausjärjestelmän ja toimintajärjestelmän liityntään
kvalifiointi	tuotoksen kelpoisuuden tarkastaminen
käyttöautomaatio|	kaikki automaatio joka ei asetu TLJ-käsitteen alle
MES	Manufacturing Execution System, tuotannonohjausjärjestelmä
mission-critical	tehtäväkriittinen tai turvallisuuskriittinen
MTBF	Mean Time Between Failure, vikaantumisväli
MTTF	Mean Time To Failure, vikaantumisaika
MTTR	Mean Time To Repair, toipumisaika
PLC	Programmable Logic Controller, ohjelmoitava logiikka
OASP	ohjeellisen automaation sovellussuunnittelun prosessimallista käytetty lyhenne
popup	käyttöliittymävuorovaukutteinen valikko
ROPES	Rapid Object-oriented Process for Embedded Software, kts. Harmony
RUP	Rational Unified Process, Rational:in yhtenäistetty prosessi

safety-critical	kts. mission-critical
SAT	Site Acceptance Test, hyväksymistestaus
SC1..4	Safety Class, turvallisuusluokka
SEPG	Software Engineering Process Group, ohjelmistoprosessin ohjausryhmä
SIL1..4	Safety Integrity Level, turvallisuuden eheystaso
tagi	I/O porttikohtainen nimetty muuttuja
TLJ	turvallisuuteen liittyvä järjestelmä
UML	Unified Modelling Language, olio pohjaiseen ohjelmiston mallinnuskieli
white box	kts. glass box

1. JOHDANTO

Yrityksen slogan *lupa luottaa* näkyy erittäin hyvin Insta Automation Oy:n pitkäaikaisten asiakkaiden vahvana luottamuksena. Tätä diplomityötä kirjoittaessa yrityksellä tulee täyteen 20 vuotta vesihuollon automaatiototeutusten ja elinkaaripalveluiden parissa. Instan asiakkailla ei ole esiintynyt tarvetta vaatia pitkäaikaiselta automaatiotoimittajaltaan erityistä laadun todistamista, jolloin varsinaista laadunvarmistusprosessia ei ole tarvinnut korostaa yrityksen automaation sovellussuunnittelun yhteydessä. Tämän diplomityön myötä kirjataan sovellussuunnittelun työprosessi yrityksen laadunhallintajärjestelmään, johon sisältyy työn aiheen mukainen kehitysehdotus suunnittelun laadunvarmistusprosessia varten. Päätavoitteenaan tämän työn pyrkimyksenä on vastata kohdeyrityksen asiakkaiden laatuvaatimuksiin, joita syntyy uusien asiakkuuksien ja näiden mukanaan tuomien vaatimuksien myötä.

1.1 Työn aihealue ja tavoitteet

Automaation sovellussuunnittelu käsittää tämän työn kontekstissa käyttöautomaation sovellusten kehittämisen. Termillä käyttöautomaatio viitataan tuotantoa jatkuvasti ohjaavaan järjestelmään eli kyseessä eivät ole turvallisuuteen liittyvät järjestelmät, jotka puuttuvat vasta havaituissa virhetilanteissa prosessin hallintaan. Seuraavaksi selitetään aihealueeseen liittyviä käsitteitä sekä kohdennetaan aiheen aluetta käsitteiden avulla. Laitteistonäkökulmasta kehitettävä sovellus suoritetaan automaatiolaitteistossa, jonka merkityksenä on kohdeprosessin automatisointi automaatioon kytkettyjen toimilaitteiden osalta. Järjestelmä muodostuu toimilaitteita ohjaavasta ja prosessia tarkkailevasta I/O-rapinnasta (tulo/lähtö), väylästä, sekä ohjelmoitavan logiikan keskusyksiköstä (CPU). Koska sovellussuunnittelijan tehtävänä on myös kohdeprosessin operointiin tarkoitetun käyttöliittymän (HMI) ohjelmointi, muodostuu HMI:stä tässä keskeinen osa työn kontekstia.

Tämän työn automaation sovellussuunnittelu -käsitteen ulkopuolelle on rajattu turvallisuuteen liittyvät järjestelmät, ylemmän tason automaation informaatiojärjestelmät (MES, ERP), reaaliaikajärjestelmät (ESW) sekä hajautetut automaatiojärjestelmät (DCS). Syyt näiden ulkopuolella rajaamiseen ovat ilmeiset: suurin osa kohdeyrityksen projekteista toteutetaan ohjelmoitavalla logiikalla (PLC). Ylemmän tason automaatiosovellukset ja osin reaaliaikajärjestelmät kehitetään hyödyntämällä ohjelmistoteollisuuden suunnittelukonsepteja.

Käyttöautomaation laadunvarmistaminen on ongelmallista, koska selkeätä toimintatapaa eikä käsitteistöä ole tarjolla. Terminologia automaation laadunvarmistuksessa on kirjavaa vaikeuttaen projektin osapuolien yhteistyötä [1, s. 13]. Turvallisuuskriittisten järjestelmien osalta tilanne on tarkemmin määriteltyä ja aihetta käsitellään useiden eri standardien muodossa yleisistä turvallisuusmäärittelyistä aina toimialakohtaisille tasoille asti. Koska TLJ-kehitys tapahtuu näiden standardien mukaisilla toimintatavoilla, ei aihetta ole syytä käsitellä tässä tarkemmin. TLJ standardeissa toistuvat laadunvarmistuksen kannalta samat periaatteet joita on tämän työn osalta tarkasteltu siitä näkökulmasta, että voitaisiinko niitä hyödyntää tämän työn kontekstiin. Kuitenkaan ei ole tarkoituksenmukaista tuoda laadunvarmistustoimia suoraan turvallisuuskriittisistä suojausjärjestelmistä koko laajuudessaan sovellettavaksi käyttöautomaation pariin, ja näin generoida liiallista kankeutta raskailla laadunvarmistustoimilla.

Kohdeyrityksen sovellussuunnittelun laadunvarmistamisen kehittämisen osalta tavoite suuntautuu mahdollisimman kevytrakenteiseen prosessiin, joka joustaa projektien vaihtelevien luonteiden mukaan. Suunnittelu- ja laadunvarmistusprosessi tulee soveltua kohdeyrityksen eri toimialojen projekteihin, jotka voivat olla laajuudeltaan suuria tai pieniä. Suunniteltavan sovelluksen laajuus riippuu suoraan automaatioon kytkettyjen laitteiden määrästä. Myös monipuoliset toiminnallisuudet vaativat laiteresursseja omalta osaltaan ja laajuuteen varaudutaankin jo laitteistoa suunniteltaessa. Sovellusta kehitettäessä sen kuluttamiin resursseihin ei voida toteutusvaiheessa täysin vaikuttaa, johtuen suunnittelun luonteesta käyttää tuotetekehityksen kehittämiä ohjelmistomoduuleja kiinteine vaatimuksineen.

Päällimmäisenä tavoitteena on perehtyä sovellussuunnittelun toimintamalleihin sekä näissä esiteltyihin laatutoimiin. Näiden perusteella pyritään muodostamaan selkeä ja johdonmukainen sovellussuunnittelun prosessin kuvaus, joka sisältää ehdotuksen laadunvarmistusprosessin kattavuuden parantamisesta. Myös varsinaista työprosessia on muokattu kirjallisuudessa esitettyjen mallien pohjalta, jotta laadunvarmistus voidaan kohdentaa sekä toteuttaa kattavammin. Sivuvaikutuksena esitettyjen muutosten myötä kohdeyrityksen sovellussuunnittelu myös nousee täysin uudelle tasolle, jossa ei pelkästään laadunvarmistus mahdollistu tehokkaammin vaan myös resurssien käyttöä voidaan tehostaa muutosten pysyessä kohtuullisen pieninä. Lisäksi kehitysehdotuksen myötä mahdollistuu uusi laatumittari, joka liittämällä olemassa olevien mittarien joukkoon tehostaa kehityksen seuraamista.

Työn tavoitteiksi asetettiin alusta lähtien kohdeyrityksen logiikkaohjelmoinnin työprosessin kuvaaminen kaikilta toimialasegmenteiltä. Tämän työprosessin kehittämiseksi siinä jo oleva laadunvarmistuksen prosessi asetettiin käsittelyn kohteeksi kirjallisuudessa esitetyistä näkökohdista. Työn aikana organisaation kypsyyden kehittämistä käsittelevien teosten asioita esitellään näihin liittyvien lukujen yhtey-

dessä. Aluksi mukana olivat myös turvallisuuskriittisten järjestelmien käsittely sekä sovellusprojektien tukitoimintojen kuvaaminen, kuten dokumentointi, vaatimustenkäsittely sekä muutostenhallinta. Työn laajuus huomioiden viimeksi mainittuja asioita käsitellään sovellussuunnittelun näkökulmista kirjallisuudesta muodostetun tiivistelmämateriaalin avulla pintapuolisesti.

Kokonaiskäsityksen muodostamiseksi työssä esitellään sovelluksen laatutekijöitä sekä laadukkaan sovelluksen tekemiseen vaikuttavia muita asioita. Suunnitteluprojektissa käytettävät tukitoiminnot laadunvarmistamisen lisäksi esitellään teoriaosuudessa. Kuitenkaan projektinhallinnallisia asioita ei käsitellä. Työssä hyödynnetään organisaation kypsyystason kohottamista käsittelevää kirjallisuutta. Koska koko organisaation kypsyystason kehittäminen on hyvin laaja prosessi jolla muokataan erityisesti organisaation projektinhallintaa, ei kypsyystason kohottamiseen liittyviä asioita käsitellä laadunvarmistusta laajemmalta alueelta.

Automaatiosovelluksen suunnitteluprosessin kuvaamiseksi haastateltiin yrityksen asiantuntijoita. Haastattelujen aikana kerätystä yksityiskohtaisesta sekä toimialakohtaisesta materiaalista on muodostettu yleinen prosessimalli, jonka pohjalle laadunvarmistuksen kehitysehdotus muodostetaan. Haastattelujen yhteydessä esiintynyttä yksityiskohtaista tietoa on käytetty lisämausteena kohdeyrityksen työprosessia kuvattaessa, vaikka yksityiskohtaisen prosessikuvauksen muodostaminen on pääasiassa rajattu ulkopuolelle laaja-alaisuudestaan johtuen. Haastatteluista kävi ilmi, että projektikohtaisista eroista aiheutuu suuria poikkeavuuksia sovellussuunnittelun kulkuun jo toimialasegmenttien sisällä. Kuitenkin eri toimialoilla esiintyvien käytänteiden erot voidaan jättää muodostettavassa mallissa huomioimatta tarkastelemalla asioita tarpeeksi korkealta abstraktiotasolta.

Osana tätä työtä toteutettu ohjeellinen automaation sovellussuunnittelun prosessimalli (käytetään lyhennettä OASP) on suunnattu automaatiosuunnittelua aloittelevien uusien suunnittelijoiden hyödynnettäväksi. Se tarjoaa organisaation toimintajärjestelmän kautta viitoitusta tuleviin projekteihin, mutta kehitysehdotuksen mukainen laadunvarmistusprosessin käyttöönottaminen on rajattu tapahtumaan tämän työn ulkopuolella. Myös kokeneemmat suunnittelijat voivat laajentaa tietämystään ohjelmistotuotannosta lähtöisin olevien käsitteiden ja käytänteiden avulla.

1.2 Tutkimusmenetelmät

Tutkimuksen perusteluna tässä työssä käytetään systemaattisen työprosessin sekä tähän liitetyiden laadunvarmistustoimien tuomia käytännön hyötyjä, jotka tulevat ilmenemään kohdeyrityksen sovellussuunnittelun laatupoikkeamien vähentymisenä. Työn lähtökohtana pidetty tutkimusongelma koostuu kahdesta aiheesta: kohdeyrityksen sovellussuunnittelun työprosessin selvittämisestä sekä työprosessiin liittyvän laadunvarmistuksen prosessin kehittämisestä.

Kohdeyrityksessä sovellussuunnittelun työprosessi on muodostunut ajan saatossa vastaamaan toimialakohtaisia vaatimuksia. Työprosessiin liittyvän laadunvarmistusprosessin identifioiminen ja kehitysehdotuksen muodostaminen edellyttävät varsinaisen työprosessin työnkulun selvittämistä. Työn kulku on alalla työskentelevillä henkilöillä hyvin tiedossa, mutta sitä ei ole aikaisemmin dokumentoitu.

Tässä esitettyä tutkimusongelmaa lähestytään seuraavilla tutkimuskysymyksillä:

TK1. Mistä toimenpiteistä muodostuu kohdeyrityksen sovellussuunnittelu?

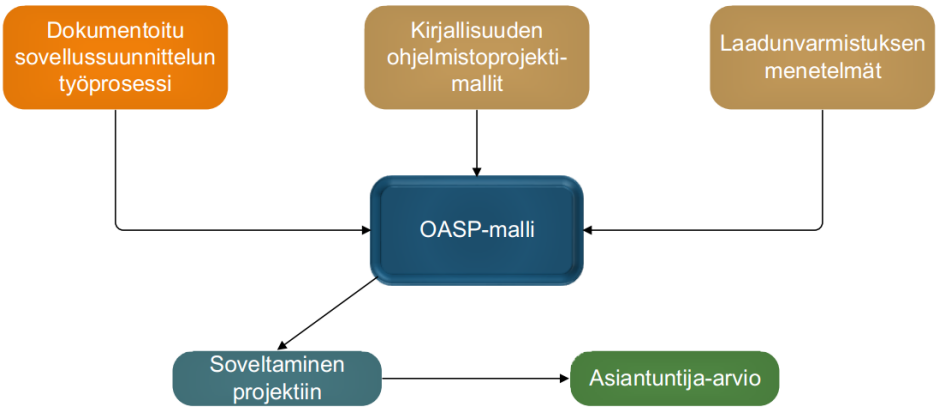
TK2. Kuinka kehitettävän automaatiosovelluksen laadusta voidaan varmistua?

Tutkimusmenetelminä käytettiin haastattelututkimusta, kirjallisuustutkimusta ja toimintatutkimusta. Lisäksi työn aikana pyrittiin reflektion avulla luomaan kokonaiskuva diplomityön aihealueesta ja alueeseen vaikuttavista tekijöistä. Molempiin tutkimuskysymyksiin kerättiin aineostoa asiantuntijahaastattelujen avulla. Haastatteluissa hiljainen tieto kirjattiin ylös strukturoimattomien kysymyksien avulla, joiden asetteluun hyödynnettiin diplomityön tekijän empiiristä työkokemusta kohdeyrityksessä. Työn tuloksena dokumentoitua suunnitteluprosessia varten kehitettiin laadunvarmistusprosessia työn tekemisen aikaisen reflektion avulla. Reflektion mukaista uuden tiedon muodostamista täytyi hyödyntää, koska täysin työn kontekstin mukaista kirjallisuutta on vähän tarjolla.

Toimintatutkimuksessa tarkastellaan empiirisesti reaalimaailmaa ja tutkija osallistuu organisaation arkipäivään. Sille ominaista on käytäntöihin suuntautuminen ja tutkimuskohteen kehittäminen muutoksen avulla. Toimintatutkimuksella pyritään kehittämään kohteena olevaa organisaatiota sen toimintatapoihin vaikuttamalla [15]. Tämän työn aikana kehityksen käyttöönottoa ei siis suoritettu. Reflektiossa yksilö selvittää kokemuksiansa uuden tiedon konstruoimiseksi tai uusien näkökulmien löytämiseksi. Reflektiivisessä prosessissa pyritään oppimiseen teoriaa ja käytäntöä ymmärtämällä [25]. Koska erityisesti käyttöautomaation sovellussuunnittelua koskevaa laadunvarmistuksen kirjallisuutta havaittiin olevan vähän tarjolla, tuli diplomityössä yhdistää tarjolla olevan kirjallisuuden menetelmiä käytännön kautta opittuihin asiayhteyksiin.

1.3 Työn rakenne

Työ rakentuu asiantuntijahaastattelujen avulla kerättyyn aineistoon ja kirjallisuustutkimuksen sekä työkokemuksen kautta saatuun ymmärrykseen. Teoriaosuuden kantavina teoksina käytetään Suomen Automaatioseura Ry:n julkaisuja, professori Ilkka Haikalan ohjelmistotuotantoa käsitteleviä kirjoja, sekä tieteellisiä julkaisuja jotka



Kuva 1.1 *Diplomityön rakenteen muodostavat tekijät.*

käsittelevät ohjelmistoteollisuutta, laadunvarmistusta sekä turvallisuuskriittisiä järjestelmiä. Erityisesti ohjelmistoprojektimallien osalta ohjelmisto- ja automaatioalan kirjallisuutta on tutkittu tarkoin. Kuvassa 1.1 on esitetty työn rakenne pääpiirteittäin.

Kirjallisuustutkimuksen myötä työhön on sisällytetty tiivistelmä sovellussuunnittelun aihealueesta, sekä eräistä projektinhallinnan tukitoimista osana asetettuja tavoitteita. Elinkaarimalleja käsittelevässä luvussa esitellään automaation elinkaari, jotta muodostuisi kokonaiskuva josta on havaittavissa automaation sovellussuunnittelun sijoittuminen teollisuuden automaatioprojektissa. Turvallisuuskriittisiä järjestelmiä, niiden toimintaympäristöä ja termistöä esitellään pintapuolisesti, kuitenkin laadunvarmistukseen liittyvien seikkojen osalta. Näihin tutustuminen oli olennainen osa kirjallisuustutkimusta soveltuvien toimintatapojen tunnistamisessa.

Laadunvarmistusprosessin kehittämiseksi hyödynnetään teoriaosuuden tarjoamia malleja ja menetelmiä. Näistä on koottu kohdeyritykselle kehitysehdotus, jota noudattamalla kohdeyritys kykenee varmistamaan tuotteensa laadun entistä paremmin. Työn loppupuolella analyysiosiossa tutkimus keskittyy sovellussuunnitteluprosessin määrittämiseen kootun aineiston avulla sekä tunnistettujen toimien implementoimiseen ohjeellisen prosessimallin (OASP) muodostamiseksi. Tätä muodostettua mallia sovelletaan erääseen projektiin ja mallia sekä sen suunniteltua soveltuvuutta arvioidaan asiantuntija-arvionnin avulla (kuva 1.1).

2. SOVELLUSSUUNNITTELUPROSESSI

Tässä luvussa esitellään sovelluksen tuottamiseen liittyvää prosessia. Ensin perehdytään suunnitteluprosessin konseptiin ja tämän jälkeen käsitellään ohjelmistoteollisuuteen kehitettyjä sovellussuunnittelun prosessimalleja. Näitä geneerisiä malleja käytetään eri lähestymistapojen kuvaamiseen sovellusten kehittämisessä. Mallit ovat siis yksinkertaistettuja esityksiä suunnitteluprosessin aktiviteeteista. Kaikkiin erilaisiin suunnitteluprosesseihin lukeutuu seuraavat neljä aktiviteettia: sovelluksen spesifioiminen, sovelluksen suunnittelu ja suunnitelman implementoiminen, sovelluksen kelpoistaminen sekä sovelluksen evoluutio eli ylläpidon aikainen kehittymä.

Edellä kuvatut aktiviteetit ovat monimutkaisia ja sisältävät useita tehtäviä sekä tukitoimia. Näitä sovellusprojektiin liittyviä tukitoimia käsitellään kursorisesti luvussa 5. Sovellussuunnittelun prosessit ovat kuten kaikki tiedolliset ja luovat prosessit jotka rakentuvat ihmisten harkintakykyyn ja päätöksiin. Tämänkään takia ei ole olemassa mitään yhtä oikeaa tapaa tuottaa sovelluksia, joten eri organisaatiot ovat kehittäneet omat prosessinsa sovellusten suunnitteluun. Vaikka ideaalia prosessia ei olekaan olemassa, alalla toimivissa yrityksissä käytettävää prosessia voidaan aina kehittää eteenpäin. Yrityksen sovellussuunnitteluprosessia voidaan viedä eteenpäin standardisoimalla toimintatapoja. Siitä muodostuu ensimmäinen askel uusien sovellussuunnittelun menetelmien käyttöönottoon sekä hyvien insinöörikäytänteiden hyödyntämiseen [39, s. 29].

2.1 Prosessiajattelu

Prosessin voidaan luonnehtia olevan systemaattinen toimintamalli tavoitteen saavuttamiseksi, joka koostuu ihmisistä, tehtävistä, järjestelmistä, työkaluista ja menetelmistä, sekä toisista prosesseista. Sovelluskehityksen toimintamallin avulla ohjelmistotuotteen laatu ei ole sattumaa, vaan täysin ennustettavissa oleva, käytettyjen menetelmien lopputulos. Prosessimallin ohjeistavat prosessin aktiviteetit, toimijat, sekä näiden vastuut. Prosessiajatteluun liittyy myös näkyvyys, ennustettavuus sekä toiminnan kehittäminen. Näkyvyydessä on kyse siitä, että voidaan osoittaa suoritettu tehtävä ohjeiden mukaisesti toteutetuksi. Prosessin ensisijainen tarkoitus on ennustettavuus. Siihen pyritään tekemällä asiat tarkoituksenmukaisella tavalla, suorittavista ihmisistä ja tilanteista riippumatta. Tällöin toimintaa on mahdollista mitata ja ohjata. Laadunhallintajärjestelmässä kuvataan työn kulkuun liittyviä prosesseja.

Prosessimaisessa toiminnassa toimintatapojen tulee olla järjestelmällisiä mutta byrokratia on minimoitava. Byrokratiaa kuitenkin lisäävät hankkeen koko, henkilöiden vaihtuvuus, henkilöiden kokemus ja ammattitaito, ylläpitovaatimukset ja järjestelmän arvioitu käyttöaika sekä tuotteen luotettavuusvaatimukset. Näihin vaikuttavat asiakkaan vaatimukset ja sovelluskohteeseen kohdistetut viranomaismääräykset ja lait [41].

2.2 Prosessin kehittäminen

Sovellussuunnitteluprosessin eteenpäin kehittäminen toteutetaan pienin askelin, sillä kehityshankkeiden eteenpäin vieminen projektipaineiden alla on hankalaa. Kehityshankkeissa ongelmaksi nousee aikaansaatujen positiivisten muutosten vakiinnuttaminen. Prosessin kehittämisen askeleet koostuvat nykytilanteen kartoittamisesta, pienien muutosten suunnittelusta, kehitystarpeiden priorisoinnista ja ensimmäisten askelien ottamisesta.

Ensimmäisten askelten joukkoon kuuluu uuden toimintaprosessin mallintaminen jokaisen vaiheen osalta. Ohjelmistoprojektissa usein lähdetään liikkeelle kehittämällä dokumentointikäytänteitä. Yleisiä ongelmia kehityshankkeissa ovat innostuksen hiipuminen alun jälkeen, liian suuret odotukset muutokseen ja rima liian korkealla, liian suuren muutoksen tekeminen kerralla, muutosvastarinta ja ajankäytön haasteet [41]. Tämän työn kohdalla suoritetaan näistä askelista nykytilanteen kartoitus asiantuntijahaastattelujen avulla, sekä pienen muutoksen suunnittelu jossa laadunvarmistus on priorisoitu.

Ohjelmistotoimittajan omien prosessien arvioimiseen ja kehittämiseen on käytettävissä *Capability Maturity Model* (CMMI) sekä tätä vastaava ISO/IEC 15504 *Software Process Improvement and Capability Determination* (SPICE) -standardit. Suuri osa ohjelmistotaloista hyödyntää CMMI-mallin mukaista etenemistä prosessiensa kehittämiseen. Näiden standardien avustamana voidaan päätellä ohjelmistoja toimittavan yrityksen kyvykkyys. Molemmissa tämä jako tehdään viiteen eri tasoon perustuvan asteikon avulla. Ne tarjoavat viitekehyksen, jonka mukaan yritys pystyy arvioimaan nykyistä kypsyystasoaan sekä tavoitteet jotka täyttämällä yritys kehittää kypsyyttään seuraavalle tasolle. Standardeissa esitetty malli käsittää menetelmät joilla auditoijat pystyvät määrittämään yrityksen sen hetkisen tason. Kypsyystasot määritellään yrityksen prosessien kontrolloinnin perusteella CMMI-mallissa seuraavasti [11]:

1. *alustava*: ohjelmistotuotanto tapahtuu ad-hoc -tyyppisesti, toimintaprosesseja ei ole määritelty, tavoitteeseen pääseminen riippuu yksilösuoritteesta, tämän tason saavuttamiseen ei ole määritelty vaatimuksia

2. *toistettava*: ohjelmistosuunnittelun kustannuksia, aikataulua ja toiminnallisuuksia hallitaan projektinhallinnan menetelmin, yritys pystyy toistamaan projektissa onnistumisensa jos on tehty samantyyppisiä aikaisemmin

3. *määritelty*: yrityksen toimintaprosessit sisältävät projektinhallinnalliset ja tekniset toiminnot standardoidusti sekä dokumentoidusti, kaikissa yrityksen ohjelmistokehityksen ja -ylläpidon projekteissa noudatetaan standardiprosessia jota pyritään tehostamaan

4. *hallittu*: ohjelmistotuotannon prosessia ja lopputuotetta mitataan erottelukykyisillä laatumittareilla ja näitä pyritään parantamaan mittaustulosten perusteella

5. *optimoiva*: kvantitatiivista prosesseista kerättyä tietoa käytetään jatkuvaan prosessien kehittämiseen ja optimointiin

CMMI-mallissa ohjelmistokehitystä käsittelevä osio jakaa ohjelmistoprojektin osiin, kuten projektin suunnitteluun tai vaatimustenhallintaan. Kypsyysarviointi tehdään prosessialueittain siten, että jokainen prosessialue arvioidaan myös viiden eri kypsyyskriteerin skaalalla. Malli pyrkii olemaan sillä tasolla yleinen, että se ei ota kantaa yrityksessä käytössä olevaan elinkaarimalliin.

SPICE on kehitetty yhdistämällä ohjelmistoalan eri malleja. Siinä prosessien kypsyys on jaettu myös viiteen eri tasoon jotka vastaavat CMMI-mallin tasoja. Myös prosessien jako eri kategorioihin on hyvin vastaava molempien mallien rakenteissa. CMMI tarjoaa hyviä käytänteitä ohjelmistokehitykseen, joita puolestaan SPICE:sta ei löydy. SPICE on yleisemmin omaksuttu käyttöön autoteollisuuden ohjelmistotuotannossa johtuen eurooppalaisen autovalmistajien asettamista vaatimuksista [7].

2.3 Sovellussuunnittelun vaiheet

Sovelluksen kehitysprosessista voidaan yleensä erottaa määrittely-, suunnittelu-, toteutus- ja testausvaihe. Vaiheita seuraa tuotteen käyttöönotto, kelpoistus sekä ylläpito. Projektiin liittyy koko projektin elinkaaren ajan kestäviä tukitoimintoja. Näitä ovat laadunvarmistus, konfiguraationhallinta, dokumentointi ja vaatimusten hallinta. Lisäksi riskienhallinta kuuluu osaksi vaaranalaisten (*mission-critical, safety-critical*) järjestelmien ohjelmistoprojektia [12].

Tässä kappaleessa esitetään ohjelmistoprojektin päävaiheet, jotka ovat yleisiä ja löytyvät kaikesta ammattimaisesta sovellussuunnittelusta. Näitä vaiheita hyödynnetään tämän diplomityön eri vaiheissa. Kohdeyrityksen dokumentoitu sovellussuunnittelun työprosessi jaotellaan haastatteluaineiston perusteella osaksi seuraavassa esitettäviä vaiheita. Lisäksi työn lopussa esiteltävä ohjeellinen sovellussuunnittelun

prosessi pohjautuu yksinomaan tässä esitettäviin sovellussuunnittelun pääveiheisiin. Ohjeellisen mallin laadunvarmistuksen prosessilla tuotettavat laadun todisteet ovat myös jaoteltu päävaiheiden mukaan. Lopuksi ohjeellisen mallin soveltaminen eräässä automaatioprojektissa laadunvarmistussuunnitelma listataan näiden tässä esiteltävien päävaiheiden pohjalta.

2.3.1 Määrittelyvaihe

Määrittelyvaiheessa toteutettava järjestelmä määritellään. Eli asiakasvaatimuksista johdetaan järjestelmävaatimukset. Nämä kattavat laitteistolle ja ohjelmistolle asetetut vaatimukset. Määrittelyn tuloksena saadaan toiminnallinen määrittely eli vaatimusmäärittely. Siinä kuvataan tuotteelta vaadittavat toiminnalliset ja ei-toiminnalliset vaatimukset. Toiminnalliset vaatimukset sisältävät järjestelmällä toteutettavat toiminnallisuudet, kommunikoinnin muiden järjestelmien kanssa sekä käyttöliittymän. Lisäksi toiminnallisessa määrittelyssä huomioidaan esimerkiksi tietyn laitevalmistajan laitteiden asettamat rajoitukset. Määrittelyvaiheessa asiakasvaatimukset muunnetaan täsmällisiksi ohjelmisto- ja laitevaatimuksiksi [12].

Automaatiojärjestelmää määriteltäessä selvitetään projektin tarpeellisuus ja toteuttamiskelpoisuus, asetetaan tavoitteet ja vaatimukset, sekä laaditaan ratkaisumalli. Automaatiojärjestelmä kuvataan käyttäjän kannalta yleiseltä tasolta. Abstrahoitu kuvaus ei ota kantaa toteutuksen teknisiin yksityiskohtiin. Määrittelyvaihetta edeltää esitutkimus, jonka tarkoituksena on tehdä päätös projektin aloittamisesta. Esitutkimuksen aikana asetetaan yleiset järjestelmätason vaatimukset, joita kutsutaan asiakasvaatimuksiksi. Esitutkimuksen tulos vastaa kysymykseen miksi järjestelmä tulisi tai ei tulisi tehdä. Esitutkimus ajatellaan osaksi määrittelyvaihetta, koska asiakastarpeiden tarkentaminen jatkuu esitutkimuksesta lähtien läpi määrittelyvaiheen. Vaihe jakaantuu kahteen osaan joista esisuunnittelu on asiakkaan vastuunalainen vaihe. Tämä pitää sisällään aineiston kokoamisen investointipäätöstä varten. Esisuunnitteluvaiheeseen kuuluu myös selvitys turvallisuuteen liittyvän järjestelmän tarpeesta [1, s. 36],[12].

Projektin perustamisen jälkeisessä perussuunnitteluvaiheessa määritellään prosessin ajotavat sekä esitutkimuksen yleisten vaatimusten mukaiset automaatiotoimintojen tarkemmat kuvaukset. Toimittajan tekemä tarjous sisältää automaatiojärjestelmälaitteiston ja ohjelmiston rakenteen toiminnallisen kuvauksen. Siinä yksittäiset toiminnot ovat kuvattu lyhyesti, sekä kerrottu kuinka nämä vastaavat asiakkaan käyttäjävaatimuksiin. Tässä tilanteessa haastavaa on yhteisymmärrykseen pääseminen asiakkaan todellisista tarpeista [12]. Neuvotteluiden aikana tarkennetaan teknisiä ja laadunvarmistuksen vaatimuksia [1, s. 18].

Määrittelyvaihe muodostuu asiakasvaatimusten kartoittamisesta ja toteutettavan järjestelmän määrittelystä. Asiakasvaatimusten kartoittamista kutsutaan usein ai-

kaisemmin mainituksi esitutkimukseksi. Määrittelyvaiheessa voidaan toteuttaa koko järjestelmän määrittely tai pelkästään jonkin osa-alueen määrittely kuten laitteiston tai ohjelmiston määrittely. Tämän vaiheen tuloksena syntyy toiminnallisen määrittelyn dokumentaatio. Tämä sisältää vaatimukset sekä kuvauksen vaatimukset täyttävästä järjestelmästä [12].

2.3.2 Suunnitteluvaihe

Suunnitteluvaiheessa asiakkaan vaatimusten mukaan tehty toiminnallinen määrittely, vaatimusmäärittely, muunnetaan järjestelmän toteutuksen kuvaukseksi. Ohjelmistoteollisuudessa tätä kutsutaan tekniseksi määrittelyksi. Sovelluksen suunnittelua koskeva suunnitteluvaihe koostuu kahdesta pääosasta arkkitehtuurisuunnittelusta ja moduulisuunnittelusta. Arkkitehtuurisuunnittelussa järjestelmä kuvataan moduuleista rakentuvana kokonaisuutena. Yksinkertaisimmillaan arkkitehtuurisuunnittelussa on kyseessä moduulien välisen työnjaon ja rajapintojen suunnittelu. Moduulisuunnittelussa suunnitellaan jokaisen moduulin toteuttavien prosessien ja tehtävien rakenne. Arkkitehtuurin voidaan ajatella myös olevan sellainen osa ohjelmistoa, joka ei muutu normaalin ylläpidon aikana. Arkkitehtuuri voidaan käsittää järjestelmän abstraktina kuvauksena, joka on alustariippumaton. Tällöin toistuviin, saman toimialan tai kohteen, projekteihin voidaan hyödyntää aikaisempaa arkkitehtuurisuunnittelua [12],[11].

Sellaisia suunnitteluperiaatteita joilla saavutetaan selkeitä ja ymmärrettäviä ratkaisuja ovat yksinkertaisuus ja suoraviivaisuus, osittaminen, lokaalisuus, abstraktioiden hyödyntäminen, rajapintakeskeisyys sekä yhdenmukainen toteutusfilosofia. Toteutusvaihtoehdoista olisi aina pyrittävä löytämään suoraviivaisin ja yksinkertaisin vaihtoehto. Toteutuksen monimutkaisuuden hallintaan vaikutetaan arkkitehtuurisuunnittelun avulla. Arkkitehtuuriltaan ositetun sovelluksen mahdolliset virheet havaitaan helpommin ja varmemmin testaamisen yhteydessä. Koodausvirheen tilanteessa ohjelman toimivuudet eivät rikkoudu tähän vuorovaikutuksessa olevien moduulien osalta. Onnistunut osittaminen on tärkeää sovelluksen selkeyden takia sekä sellaisten laatuominaisuuksien, kuten luotettavuuden, ylläpidettävyyden ja uudelleenkäytettävyyden kannalta [12].

Suunnitteluperiaatteista abstraktioilla tarkoitetaan ainoastaan keskeisen tiedon esittämistä, jolloin detaljitieto on piilotettu. Abstraktioiden hyödyntäminen lisää siis suunnittelun ymmärrettävyyttä joka syntyy yksityiskohtien piilottamisesta ja rajapinnan selkeydestä. Myös yhdenmukainen toteutus läpi kehitettävän sovelluksen parantaa omalta osaltaan ymmärrettävyyttä. Arkkitehtuurisuunnittelun tavoitteena on suunnitella moduulit niin, että ne ovat toisistaan riippumattomia. Lisäksi arkkitehtuurisuunnitteluun kuuluvassa rajapintamäärittelyssä moduulien rajapinnat tulisi määritellä mahdollisimman sovelluskohderiippumattomasti. Näin toteu-

tetut moduulit voidaan kehittää toisistaan erillään, sekä kehitettyä ohjelmalohkoa voidaan paremmin hyödyntää myöhemmissä projekteissa [12].

Automaatiosovelluksen suunnittelu muodostuu perinteisesti dokumenttipohjaisesta suunnittelusta. Siinä suunnitteluvaiheen lähtötietojen dokumentteja ovat laitesekä I/O-luettelo, prosessi-instrumentointikaavio, järjestelmän toimintakuvaus, säätökaaviot, kokousmuistiot ja muut projektin tekniset dokumentit. Yhteen dokumenttiin tullessa muutoksia voi tämä vaikuttaa useisiin muihin dokumentteihin. Tällöin dokumenttien ylläpito muodostuu hankalaksi ja päivittämättömät dokumentit vanhenevat käyttökelvottomiksi nopeasti. Kuitenkaan kaikkien dokumenttien ylläpito läpi projektin ei ole tarkoituksenmukaista. Projektin aikana ilmenevien muutosten tuomiin haasteisiin vastataan muutostenhallinnalla. Ylläpidettävien dokumenttien osalta tuleekin huolehtia näiden helposta ylläpidettävyydestä minimoimalla tarpeettomien yksityiskohtien kirjaaminen helpoisti muuttuvien tietojen osalta (I/O-osoitteet).

Dokumenttipohjaisessa suunnittelussa vaatimusten jäljitettävyys on vaikea ja työläs toteuttaa. Tähän tuo apua mallipohjainen suunnittelu, jossa yksittäisiä dokumentteja ei tarvitse päivittää käsin, vaan muutokset tehdään järjestelmää kuvaavaan malliin. Erilaisten näkymien avulla mallia voidaan tarkastella eri näkökulmista ja suunnitellusta mallista voidaan mallinnustyökalun avulla generoida yleisesti käytettäviä dokumentteja.

Automaatiosovellusten mallintamiseen käytettävät IEC 61131-3 ja IEC 61149 standardien tarjoamat mallinnuskielet mahdollistavat toteutuksen detaljisuunnittelun, mutta korkeamman tason suunnitteluun kuten arkkitehtuurisuunnitteluun ne eivät sovellu. Ohjelmistoteollisuudessa yleisesti käytössä olevan UML-kieli ei suoraan sovellu tämän työn kontekstin mukaiseen sovellussuunnitteluun oliokeskeisyydestään johtuen. Mallipohjaisen suunnittelun mukainen korkean abstraktiotason kuvauksen muodostaminen sovelluksesta onnistuu tällä hetkellä laajasti käytössä olevalla SysML-notaatiolla. [46].

2.3.3 Toteutusvaihe

Toteutusvaiheessa automaatiojärjestelmän toimittaja toteuttaa suunnitelmien mukaisen ohjelmiston, laitehankinnat ja keskusvalmistuksen. Toteutus tehdään toimittajan laadunhallintajärjestelmän mukaisesti. Määrittelyvaiheessa luotuihin laatutavoitteisiin voidaan vaikuttaa myös toteutusvaiheessa. Laatutavoitteisiin vaikuttavia ohjelmiston ominaisuuksia painotetaan tilanteen mukaan. Kehitysprojektissa, jossa tavoitteena on koodin mahdollisimman hyvä uudelleenkäytettävyys tiedossa olevien tai mahdollisten tulevien tilausten muodossa, tehdään täysin erilaisia suunnitteluratkaisuja verrattuna projektiin, jossa laatutavoitteeksi asetetaan mahdollisimman hyvä tehokkuus [1, s. 19]. Jos ohjelman kehitys perustuu jo olemassa olevaan pro-

jektiin, on työ luonteeltaan vanhan kehittämistä. Silloin käytetään uudelleen jo olemassa olevia komponentteja ja alkuperäisen ohjelmiston tärkeimpinä laatutekijöinä ovat ohjelmiston siirrettävyys ja joustavuus. Ne voidaan nähdä perusedellytyksinä muille laatutavoitteille, kuten luotettavuus, uudelleenkäytettävyys ja ylläpidettävyys, mutta toisaalta nämä saattavat olla myös ristiriidassa tehokkuuden kanssa [12].

Automaatiosovelluksen toteuttaminen tapahtuu määrittelydokumentaation mukaisesti. Sovellussuunnittelun reunaehtoina ovat määrittelydokumentaation mukaisten toimintakuvausten lisäksi projektin laitehankinnat. Valitut väylä- sekä I/O-liitäntäiset laitteet huomioidaan sovelluksessa. Ohjelmoitavien logiikoiden sovelluskehitysympäristöt ja näiden oheistyökaluohjelmistot määräytyvä ohjelmoitavan logiikan laitevalmistajan sekä tuoteperheen mukaan. Toteutusvaiheessa sovellussuunnittelija toteuttaa valituille laitteille sovelluksen, jonka toiminnalliset ja ei-toiminnalliset määritykset toteutuvat lähtötietojen mukaisesti. Suunnittelijalla käytettävänään ovat standardin mukaiset ohjelmointikielet sekä näille tarjolla olevat valmiit ohjelmamoduulikirjastot. Kirjastojen sisältö koostuu laitevalmistajan kehitysympäristön mukanaan tulevista standardilohkoista, yrityksen tuotekehityksen kehittämistä lohkoista sekä aikaisempiin projekteihin kehitetyistä. Näitä hyödyntämällä ja omia lohkoja toteuttamalla suunnittelija ohjelmoi oman kädenjälkensä mukaisen automaatiosovelluksen organisaation koodauskäytänteiden mukaisesti.

Sovellus testataan toiminnallisten ominaisuuksiensa ja suoristuskyvyn osalta. Testausprosessi muodostuu testaamisen suunnittelusta, testin suorittamisesta sekä tulosten analysoinnista ja raportoinnista. Ohjelmiston testaamisen tarkoituksena on havaita ohjelmistosta virheitä ja puutteita, sekä varmistaa että ohjelmisto vastaa asiakasvaatimuksiin. Testaaminen jaotellaan usein yksikkö- eli moduulitestaukseen, integrointitestaukseen ja järjestelmätestaukseen. Lisäksi automaatiosovelluksiin liittyvät asiakkaan kanssa suoritettavat kelpoistustestit kuten tehdastestaus ja hyväksyntätestaus.

2.3.4 Käyttöönotto

Käyttöönotto voidaan aloittaa kun kohdeprosessin ja automaatiotoimituksen asennukset ovat saavuttaneet tietyn valmiustason. Asennuksen aikana automaatiotoimittaja tai pääurakoitsijan hankkima asennusurakoitsija asentaa automaatiojärjestelmän kenttälaitteineen. Asennuksen jälkeen laitos on automaation osalta valmis piirikoestusta ja toiminnallisia testejä varten. Asennetun järjestelmän toiminnallisuudet testataan piiri kerrallaan kylmätestauksessa sekä lopuksi kokonaisuutena kuumatestauksessa. Kylmätestauksessa testataan yksittäiset piirit kerrallaan, sekä hälytykset, lukitukset että muut turvallisuustekijät. Testattaviin piireihin lukeutuvat kohdeprosessin mittaukset ja anturoinnit, venttiilit ja muut toimilaitteet se-

kä moottoripiirit. Automaatioon kytketyn laitteen toiminta testataan sähköisten, mekaanisten ja ohjelmallisten tekijöiden osalta. Tällöin kohdeprosessissa ohjattavia laitteita ajettaessa niitä ajetaan tyhjinä tai käytetään vaarattomia prosessiaineita kuten vettä. Tämän jälkeen kohdeprosessi tai sen osa on valmiina kuumatestaukseen, jossa kuormituksella testataan sekä prosessikemikaaleilla suoritetaan koeajo. Tämän aikana testataan kaikki automaatiosovelluksen toiminnot ja sekvenssit sekä viritetään säätöpiirit. Tarvittaessa suoritetaan erillinen hyväksyntätestaus. Vaihe päättyy järjestelmän luovutukseen, jonka jälkeen vastuu automaatiojärjestelmästä siirtyy asiakkaalle ja järjestelmän takuuaika alkaa, jos erillistä kelpoistusjaksoa ei vaadita [1, s. 20].

Kelpoistusjakso alkaa kun automaatiojärjestelmä on luovutettu asiakkaalle. Se koostuu kahdesta vaiheesta, automaation teknisestä kelpoistuksesta sekä prosessikelpoistuksesta. Tänä aikana näytetään, että laitoksella pystyy tuottamaan määritellyn mukaista lopputuotetta. Automaation tekninen kelpoistus on asiakkaan toimi johon voi kuulua erillisiä lisätestejä sekä toimittajan testiraporttien katselmointia. Näin asiakas voi varmistua siitä, että järjestelmä on suunniteltu ja toteutettu vaatimusten mukaisesti. Teknisen järjestelmän kelpoistusvaiheeseen kuuluu myös automaatiojärjestelmän osalta suorituskykytestaus. Kun loppukelpoistusraportti on hyväksytty, voidaan automaatiojärjestelmä asettaa tuotannolle kelpoistusjakson aikaisella sovellusversiolla. Tuotannon lopullinen hienosäätö tapahtuu prosessikelpoistusvaiheessa. Siinä tavoitteena on osoittaa että lopputuote on spesifikaation mukaista. Jatkuvatoimisen prosessin kelpoistus vaatii määritellyn ajan määritellyn mukaisen lopputuotteen tuottamista. Laitoksen tuotantovaiheen käynnistyessä kelpoistus on suoritettu hyväksytysti ja automaation elinkaari jatkuu ylläpidon merkeissä, jolloin automaatiojärjestelmälle suoritettavat muutokset toteutetaan muutostenhallinnan avulla. Toteutettujen muutosten myötä mahdollisesti alkaa uusi kelpoistusjakso muutosta käsittelevän kohdeprosessin osalta [1].

2.4 Automaatiosovelluksen suunnittelu

Automaatioinsinöörit ovat ensisijaisesti järjestelmä- sekä toimiala-asiantuntijoita ja vasta toissijaisesti ohjelmistoasiantuntijoita, ei ohjelmistotuotannon menetelmät ja käsitteet ole heille yleisesti tuttuja. Automaatioalalta poiketen informaatioteknologian ohjelmistoteollisuudessa on pitkät perinteet ohjelmisto- ja sovellussuunnitteluprojektimallien sekä näihin liitettyjen laadunvarmistustoimien kehittämisestä. Näitä ei ole vastaavassa määrin kehitetty automaation sovellussuunnittelua varten. Tämän ajatellaan johtuvan automaatiototeutusten suhteellisen matalista kustannuksista verrattuna ohjelmistoteollisuuden projekteihin, sekä siitä tosiasiasta kuinka pieni osuus automaatiotoimituksesta muodostuu koko teollisuusprojektin kustannusrakenteessa. Muita syitä kehityksen uupumiseen löytyy automaatioalan kon-

servatiivisuudesta sekä alan luonteelle ominaisesta projektikiireestä. Tämän takia laatua parantavat toimintamallit on otettu käyttöön automaatiosovelluksia suunnittelevissa yrityksissä myöhemmin kuin tietotekniikka-alalla [2, s. 113].

Automaatio- ja ohjausjärjestelmät käsittävät pienempien kokonaisuuksien, teollisuuslaitosten sekä koneiden ohjaamiseen käytettävät ohjelmoitavat logiikat. Tässä luvussa määritellään työn kontekstin mukainen sovellussuunnittelu sekä tehdään lyhyt katsaus automaatiosovellussuunnittelua koskevasta tutkimustyöstä sekä tulevaisuuden näkymistä. Laajempien ja yksinomaan teollisuuden prosessilaitteiston automatisointiin käytetään ohjelmoitavia logiikoita laajempia hajautettuja automaatiojärjestelmiä. Näiden lisäksi on olemassa myös PC-arkkitehtuuriin pohjautuvia automaatioratkaisuja. Ylemmän tason (ISA-95) automaatiosovellukset kattavat tuotannon ohjausjärjestelmät sekä yrityksen liiketoiminnan ohjausjärjestelmät. Nämä ovat ohjelmistotuotannon menetelmin toteutettuja sovelluksia ja ohjelmistokokonaisuuksia, joihin on tarjontaa kaikilta suurilta ohjelmistoyrityksiltä. Ohjelmistotuotannon mukaisilla ohjelmointikielillä toteutettuja sekä laiterajapinnalla vuorovaikutusta suorittavia järjestelmiä ovat reaaliaikajärjestelmät sekä sulautetut järjestelmät. Reaaliaikajärjestelmiä käytetään vaativimmissa sovelluskohteissa, joiden hallintaan tarvitaan paljon laskentatehoa. Sulautetut järjestelmät puolestaan ovat pienemmän laskentatehon sekä suurina sarjatuotantomäärinä tuotettavien laitteiden ohjaamiseen käytettäviä ratkaisuja.

Moderneissa ohjelmoitavissa logiikoissa on mahdollisuus ylläpitää web-palvelinta käyttöliittymän esittämiseksi. Tämän tyyppiset ratkaisut eivät kuitenkaan tarjoa tarpeeksi monipuolisia ominaisuuksia valvomon toteuttamiseen jotta tekniikka olisi saavuttanut laajempaa käyttöä. Aika näyttää laajentaako HTML5:n tuomat mahdollisuudet logiikkapohjaisten käyttöliittymien hyödynnettävyyttä. Tämän hetken kuumin asia esineiden Internet (IoT) on siis ollut teollisuudessa jo tätä päivää teollisuus-Ethernet standardin mukaisten laitteiden saavutettua suosion. Tämän myötä on muodostunut käsite teollisuuden IoT:stä, joka eroaa kuluttajan IoT:sta. Teollisuuden IoT yhdistää antureiden ja automaatiolaitteiden muodostaman kokonaisuuden, jolta käytetään tuotantoprosessin hallintaan. Nämä nähdään infrastruktuurina joka täytyy muodostaa, ennen kuin voidaan valmistaa kuluttajille suunnattuja IoT laitteita. Teollisuuden IoT:ssä on kyse kriittisten prosessi- ja koneenohjausten muodostamisessa. Näiden yhteydessä tapahtuvat laiterikot vaikuttavat ympäristöönsä ja ovat turvallisuuskriittisempiä, toisin kuin kuluttajille suunnattut IoT laitteet. Tämän myötä on tarjoutunut mahdollisuus innovoida uusia tapoja hyödyntää teollisuuden prosesseista kerättävää suurta määrää informaatiota tallennetuista mittaus- ja ohjaussuureista.

Sovellussuunnittelijan työkalujen ohjelmointikielet koostuvat ennalta määritellyistä ohjelmistoelementeistä, joilla tarkoitetaan esimerkiksi JA sekä TAI -loogisia

funktioita tai vaikka ajastimia. Näiden ohjelmistoelementtien avulla voidaan teollisuusprojektin vaatimukset muuttaa toiminnallisuudet toteuttaviksi logiikkakaavioksi. Työkaluohjelmistot tarjoavatkin valmiina elementteinä standardit ohjelmalohkot toimilaitteiden hallintaa I/O:n ja väylän välityksellä, tai PID-säädön toteuttamiseen tarkoitetun lohkon sekä ohjelmointimahdollisuuden omien toimilohkojen toteuttamiseen. Lohkojen sisäinen ohjelmointi voidaan toteuttaa IEC 61131-3 standardiin pohjautuvilla eri ohjelmointikielillä, joihin lukeutuvat kaksi tekstuaalista kieltä, kaksi graafista kieltä sekä sekvenssien muodostamiseen tarkoitettu ohjelmointikieli. Muodostetut lohkot integroidaan sovellukseksi yhdistämällä ne toisiinsa muuttujien avulla tai graafisella ohjelmointinäkymällä, joka voi olla CAD -tyyppinen suunnittelutyökalu jossa ohjelmalohkojen signaalit kytketään graafisten viivojen avulla. Tämän tyyppinen ohjelmointifilosofia pohjautuu standardiin IEC-61499 ja sitä hyödyntävät erityisesti hajautettujen automaatiojärjestelmien laitevalmistajakohtaiset kehitysympäristöt.

Automaation sovellussuunnittelun tieteellinen tutkimus painottuu olio-pohjaisten kehitysmenetelmien ja näihin soveltuvien ketterän kehitysmallin mukaisten toimintatapojen tuomiseen osaksi automaatiosuunnittelua, niin tuotantoa jatkuvasti ohjaavan automaation kuin nyt myös TLJ:n osalta [50]. Laadunvarmistuksen kannalta tieteellisiä tutkimuksia on julkaistu automaatiosovelluksen automaattisen todentamisen ympäristössä. Näissä tapauksissa ohjelmakoodi muunnetaan malliksi, joka todennetaan symbolisten tekniikoiden avulla [45],[4].

Tieteellisissä julkaisuissa on käytetty yksinomaan kehitystyökaluja PLCOpen-avoimesta kehitysympäristöstä. Yleisesti katsottuna useissa eri tutkimuksissa on muunnettu IEC 61131-3 standardin mukainen kieli tunnetuksi ja laajalti käytetyksi ohjelmistoteollisuuden ohjelmointikieleksi. Tällöin on saavutettu ohjelmointikielen, kuten Java tai C-kielet, ohjelmakoodin todentamiseen kehitettyjä tarkastustyökaluja esimerkiksi staattisen testauksen osalta [18]. Staattisessa testauksessa koodia tarkastetaan ilman sen varsinaista suoritusta ajoympäristössä. Tätä voidaan käyttää esimerkiksi koodin syntaksin ja muotoiluseikkojen tarkastamiseen. Tulevaisuuden näkymiä sovellussuunnittelussa edustaa mallintaminen monissa sen merkityksissä. Yhtenä näistä vaatimusten muuntaminen loogiseksi matemaattispohjaiseksi malliksi, joka mahdollistaa ohjelmakoodin automaattisen generoinnin sekä mallin pohjalta tapahtuvan automaattisen todentamisen [39].

2.5 Turvallisuuskriittinen automaatio

Automaatiosovelluksen laatuun kiinnitetään erityisesti huomiota, kun riskejä pienennetään automaation avulla. Tällaisissa sovelluskohteissa ongelmatilanteessa voi syntyä merkittäviä henkilö-, omaisuus- tai ympäristövahinkoja. Prosessiteollisuudessa tällaisiin tilanteisiin varaudutaan suojausjärjestelmillä, jotka puuttuvat proses-

sin ohjaukseen vasta vakavan häiriötilanteen sattuessa. Prosessia jatkuvasti ohjaavan käyttöautomaation osalta vaaranalaisten kohteiden selkeät toimintamallit muun muassa vaatimusten tunnistamiseen, laatutoimien kohdistamiseen sekä osapuolten vastuunjakoon ei ole määritelty [1, s. 10].

Turvallisuuskriittisiä standardeja on muodostettu kaikille turvallisuuskriittisille aloille, kuten sotilasteollisuus, ilmailuala, lääketeollisuus sekä ydinvoimateollisuus. Näissä pääpainotus on laadunvarmistuksen kannalta järjestelmän, siis suoritettavien sovellusten ja automaatiolaitteiden, kelpoistamisessa. Automaatioprojekteissa kelpoistaminen asettuu usein asiakkaan tai loppuasiakkaan vastuualueelle. Automaatiotoimittajan vastuulle jää kerätä todistusaineistoa toimitettamansa järjestelmän laadusta. Tämä tapahtuu toimittajan laadunvarmistuksen keinoin siihen hetkeen asti, kun järjestelmä hyväksytään toimitetuksi.

Lääke- ja elintarviketeollisuudessa sovellettavissa hyvissä tuotantotavoissa ohjeistetaan kuinka varmistua, että tuotantoa ja laadunvalvontaa koskevat järjestelyt ovat toteutettu turvallisuuden varmistavalla tavalla. Ydinvoima-alalla toimitaan tarkkaan määriteltyjen prosessien mukaan turvallisuuskriittisiä järjestelmiä käsiteltäessä, mutta pääasiallinen fokus ei ole sovellussuunnitteluprosessin arvioimisessa vaan laitteiston sertifioinnissa. Teknologiat joilla ydinvoimateollisuuden järjestelmät toteutetaan, pohjautuvat yleisesti teollisuudessa käytettyihin ovat usein ohjelmistopohjaisia. Tällöin järjestelmien luotettavuuden todentaminen muodostuu ongelmalliseksi johtuen ohjelmistosovellusten monimutkaisesta luonteesta [27].

Ydinvoimateollisuudessa jokaiselle uudelle järjestelmälle kohdistetaan turvallisuusvaatimusten mukainen kvalifiointi sen mukaisesti, minkä kategorian alle järjestelmä asettuu. Kvalifioinnissa on kyse kelpoistusprosessin osasta jolla osoitetaan, että järjestelmän tietyn vaiheen tuotos täyttää asetetut vaatimukset [1, s. 124]. Järjestelmät luokitellaan standardin IEC 61226 mukaisesti kolmeen kategoriaan. Näistä kategoria A vastaa turvallisuuskriittisiä järjestelmiä, eli ydinvoima-alan TLJ-järjestelmiä, kategoriat B ja C käsittävät ohjaukseen ja informaation esittämiseen liittyvät järjestelmät [44, s. 9] Kategorian A ohjelmistonäkökulmia käsitellään tähän kohdennetussa standardissa IEC 60880 ja kategorioiden B ja C näkökulmat käsitellään ohjelmistostandardissa IEC 62138 [27]. Mikäli automaatiotoimittaja haluaa toimia ydinvoima-alalla, tulee sen opetella toimimaan alalle ominaisten piiretiden mukaisesti.

TLJ sekä turvallisuuskriittisten järjestelmien luokitteluun ei ole olemassa yhdenmukaista menettelyä. Säteilyturvakeskus STUK on määritellyt ydinvoimalaitoksen turvallisuuden neljään tasoon SC1..SC4, joista SC1 on korkein taso. TLJ-järjestelmiä käsittelevä standardi SFS-EN 61508 määrittelee neljä turvallisuuden eheyden tasoa SIL1..SIL4, joista SIL4 on korkein ja vaativin. Ydinvoimateollisuuteen määritellyistä kolmesta eri kategoriasta (A, B ja C) jokaiseen voidaan yhdistää mikä tahansa

SIL-taso [14],[27].

2.5.1 Lääketeollisuus

Lääketeollisuudelle kirjoitettu *Good automated manufacturing practice* (GAMP) -ohje on tarkoitettu lääketeollisuuteen toimittaville järjestelmätoimittajille oppaaksi kelpoistettavissa olevan järjestelmän toteuttamiseksi [1, s. 11]. Vaikka ohje pääasiassa tarkoitettu lääketeollisuuteen toimittaville yrityksillä, siinä käsitellään asioita loppuasiakkaan näkökulmasta. Ohjeen tarkoituksena on auttaa suunnittelemaan järjestelmä hyviä käytäntöjä noudattaen sekä tuottamaan dokumentteja todisteeksi määrittelyjen noudattamisesta järjestelmän laadusta varmistumiseksi. GAMP laajentaa ohjeistusta tietojärjestelmiä varten. Eräs tämän tärkeimmistä asioista on ohjelmistosovelluksen komponenttien luokittelu siten, että tiettyyn luokkaan kuuluville komponenteille voidaan kohdentaa oikean laajuiset kelpoistustoimet. Sovelluskomponenttien luokittelu on hyvin vastaava kuin IAEA:n teknisessä raportissa [44] esitetään ydinvoima-alan sovelluksille.

Luokkaan 1 kuuluvat vakiintuneet kaupalliset käyttöjärjestelmät. Näitä ei tarvitse erikseen kelpoistaa, mutta ne kelpoistuvat osana sovellusta. Luokan 2 ohjelmistoihin käyttäjillä ei ole pääsyä (firmware). Näihin ohjelmistoihin tehdyt parametroinnit sisällytetään konfiguraationhallintaan. Luokka 3 muodostuu yleisesti käytetyistä kaupallisista sovelluksista, joiden kelpoistusta ei vaadita. Luokka 4 muodostuu järjestelmistä, jotka asiakas voi itse parametroida omassa sovelluskohteessaan käyttöön. Näihin kuuluvat kaikki ohjelmoitavat logiikat, hajautetut automaatiojärjestelmät sekä muut automaatiosovellukset. Järjestelmän tulee kuitenkin olla laajasti käytössä oleva ja tunnettu ennen kuin se kuuluu luokkaan neljä, muuten se lukeutuu luokkaan viisi. Luokkaan 5 kuuluu sovelluskohteeseen kehitettävät sovellukset ja tämän luokan sovellukset tulee säännöllisesti kelpoistaa koko niiden elinkaaren ajan [1, s. 187].

2.5.2 Turvallisuuteen liittyvät järjestelmät

SFS-EN 61508-standardi antaa yleisen lähestymistavan turvatoimintoja suorittaville järjestelmille. Siinä otetaan huomioon koko elinkaari turvatoimintoja suorittavan järjestelmän ja ohjelmiston osalta. Standardissa esitellään riskipohjainen määrittelytapa jota käytetään turvallisuuden eheyden vaatimusten kuvaamiseen. Turvallisuuden eheyden tasojen avulla turvatoiminnoille määritellään tavoitetasot, jotka toteutetaan TLJ:llä. Standardi asettaa myös turvallisuuteen liittyville järjestelmille tavoitteelliset vikaantumismitat jotka liitetään turvallisuuden eheyden tasoihin. Lisäksi siinä asetetaan vaatimukset systemaattisen vikaantumisen välttämiseksi teollisuudesta saavutetun kokemuksen perusteella [36, s. 13-14].

2.5.3 Ydinvoimateollisuus

Ydinvoima-alan standardit ovat kehitetty tulkitsemaan IAEA:n turvaohjeita ydinvoimalaitosten turvallisuutta toteutettavissa järjestelmissä. Näissä kiinnitetään huomiota laitteiston ja ohjelmiston keskinäiseen vuorovaikutukseen sekä erittäin luotettavan ohjelmiston kehitysprosessin yleiseen periaatteelliseen lähestymistapaan. Standardit sisältävät yksityiskohtaisia vaatimuksia sekä laadun kehittämisen valvontaan liittyviä seikkoja. Lisäksi standardeissa kuvaillaan yleinen periaatteellinen lähestymistapa ohjelmiston ja koko järjestelmän todentamiseen sekä kelpoistamiseen. Osa standardeista käsittelee formaalisti ohjelmiston ylläpitoa ja muutosten tekoa sekä konfiguraation valvontaa [14].

IAEA:n teknisen raportin *No. 384, Verification and Validation of Software Related to Nuclear Power Plant Instrumentation and Control* [44] perusteella ydinvoimaalalla käytettävät ohjelmistot ja niiden komponentit jaetaan neljään tyyppiin. Jako on tehty jotta kehitettävän sovelluksen todentamisen ja kelpoistamisen toimet voitaisiin kohdentaa tehokkaasti. IAEA:n raportissa määritellään vaadittavat toimet sen mukaisesti mihin tyyppiin sovellus lukeutuu.

Ohjelmistolle asetettavat vaatimukset tulisi johtaa turvallisuuteen liittyvän järjestelmän yleisemmistä vaatimuksista. Ohjelmistokehityksen suunnitteluvaiheen päätteeksi tuotetaan formaali dokumentti joka sisältää ohjelmiston ominaisuuksien ja suorituskyvyn spesifikaation. Dokumenttia käytetään muodollisissa suunnittelukatselmuksissa sekä suunnitteluvaihetta seuraavassa toteutusvaiheessa [14].

Ohjelmistolle tehtäviin muutoksiin tulee laatia muodolliset menettelyt, jotka koostuvat menettelyistä muutosten pyynnöille ja näiden arvioinneille, sekä menettelyt muutosten ja ylläpitotöiden läpiviemiseksi. Ohjelmistokehityksen muutoksia hallitaan muutostenhallinnan muodostavalla projektin tukitoimella (luku 5). Lyhyesti sanottuna muutospyynnöstä tulee eritellä tarve, syyt ja tavoitteet. Ennen varsinaista työn toteuttamista pitää arvioida tehtävän muutoksen aiheuttamat vaikutukset kokonaisuuteen sekä laitteistoon että ohjelmistoon, muutosten tekniseen soveltuvuuteen ja niiden realisoituvuuteen. Menettelyiden laajuus riippuu elinkaaren vaiheesta, jossa muutokset ja ylläpitotyö toteutetaan [14].

3. ELINKAARIMALLEJA

Ohjelmistoprojektien elinkaarimallin tehtävänä on ohjata sovellussuunnitteluprojektin etenemistä. Sovellussuunnittelussa käytettävät elinkaari- ja ohjelmistoprojektimallit tarjoavat työntekijöille ohjausta työn etenemiseen sekä projektin, joka on ennustettavissa niin aikataulun kuin vaadittavien resurssienkin suhteen. Hyvää elinkaarimallia noudattamalla ohjelmistotuotteen laatu paranee oikeellisuuden kasvamisena, virhetoimintojen vakavuuden alentumisella, parempana uudelleenkäytettävyydellä ja paremmalla ylläpidettävyydellä [6].

Elinkaarimallissa sovellussuunnittelun kehitys esitetään korkean abstraktiotason esityksenä jossa kehitysprosessi tapahtuu tietyssä järjestyksessä suoritettavina vaiheina. Mallit esittävät ohjelmistoprojektin merkittävimmät tapahtumat, sekä siirtymäkriteerit näiden välillä. Niiden avulla määritellään mitä, miten ja milloin tehdään. Tavallisimpia malleja ohjelmistotuotannossa edustavat vaihejakomalleihin perustuvat, iteratiiviset ja ketterät lähestymistavat [11].

Yksinkertaisimmassa lähestymistavassa sovelluksen kehitys tapahtuu sovelluksen koodia lisäämällä ja korjaamalla, kunnes lopputulos on tyydyttävä [12]. Tämän seurauksena koodin rakenne muuttuu helposti sekavaksi ja laatukriteerit heikkenevät merkittävästi (tehokkuus, joustavuus, ylläpidettävyys jne.). On myös huomattu että heikosti jäsennellyn koodin muuttaminen on aikaa vievää ja tätä kautta kallista. Tämän välttämiseksi tarvitaan systemaattinen suunnitteluprosessi, jossa esiintyy määrittelyvaihe jotta sovellus vastaa paremmin asiakkaan toiveita sekä suunnitteluvaihe ennen ohjelman koodaamista [3].

3.1 Automaation elinkaari

Automaation elinkaarella tarkoitetaan aikaa automaatiosuunnittelun alkamisesta tilanteeseen jossa automaatiojärjestelmä poistetaan käytöstä. Suomen Automaatioseura Ry:n julkaisussa *Laatu automaatiossa - Parhaat käytännöt* [1] esitellään automaation elinkaarimalli kuvassa 3.1. Tässä esityksessä automaation elinkaaresta keskitytään teollisuusprojektin automaation toimittajaa käsitteleviin osuuksiin. Kirjassa käsitellään automaatiosovellussuunnittelun laadunvarmistustoimia. Näillä toimilla varmistetaan projekteja, jotka sisältävä vaativia järjestelmähankintoja prosessiin liittyvien vaaratekijöiden takia.

Automaation elinkaarimallissa on esitetty keskeiset päätöksentekotilanteet etapit,

etappien väliset elinkaarivaiheet, elinkaarivaiheen kannalta tärkeäksi luokitellut tulokset, sekä vaiheisiin liittyvät laadunvarmistustoimet. Elinkaarimallissa laadunvarmistus tapahtuu automaation sovellussuunnittelun kannalta asiakkaan kanssa suoritettavien katselmuksien myötä, ja toteutuksen sekä toiminnallisen testauksen aikana suoritettavan testaamisen kautta. Laatu automaatiossa -kirjassa laadunvarmistuksen periaatteeksi mainitaan etukäteen suunniteltu laatu, sovelluskehityksen aikana suoritettu kattava dokumentaatio sekä todentamisen ja kelpoistamisen menetelmin tapahtuva tuloksien vertaaminen lähtötietoihin [1, s. 122].

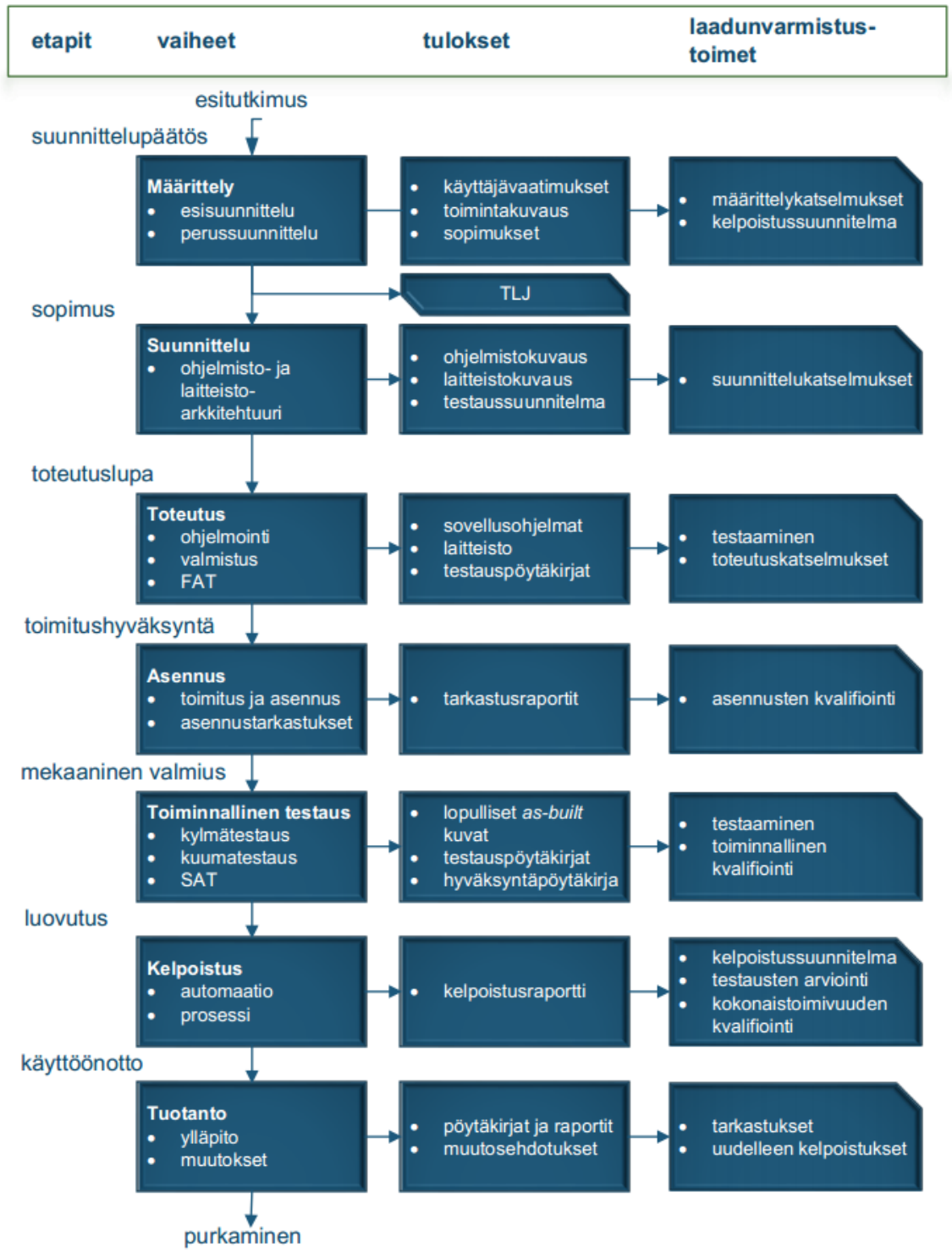
Tyypillisiä sovellussuunnittelijan vastuualueelle kuuluvia työvaiheita ovat elinkaarimallissa sopimuksen jälkeen tapahtuva suunnittelu, sovelluksen toteutus, sekä järjestelmän käyttöönotto. Vaiheiden tarkempaan sisältöön tutustutaan myöhemmissä luvuissa. Elinkaarimallissa jokaista vaihetta vastaa tähän kohdennettavat laadunvarmistuksen toiminnot. Näihin lukeutuvat automaatiosovelluksen tapauksessa testaaminen ja katselmukset. Laadunvarmistusosuudessa esiintyvällä kvalifioinnilla tarkoitetaan asiakkaan suorittamaa kyseisen vaiheen tulosten kelpoisuuden tarkastamista.

3.2 Vesiputous- ja v-malli

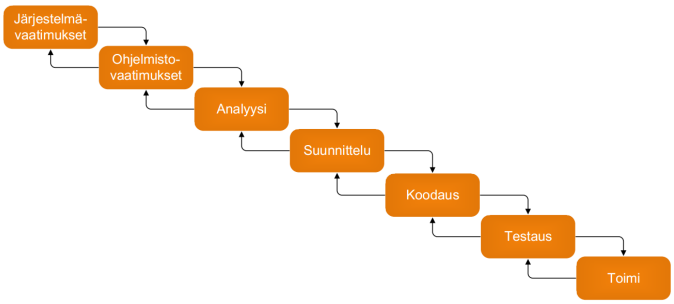
Vesiputousmallissa sovelluksen suunnitteluprosessi on jaettu peräkkäin suoritettaviin vaiheisiin. Malli pohjautuu vuoden 1970 julkaisussa *Managing the Development of large software systems* [34] esitettyyn näkemykseen, jossa sovellussuunnittelun varsinaista toteutusta edeltää kaksivaiheinen vaatimusanalyysi ja toteutuksen jälkeinen sovelluksen testaus. Vesiputousmalli on saanut alkunsa julkaisussa esitetystä prosessimallin rungosta, joka esitetään kuvassa 3.2. Julkaisussa esitellään iteratiivinen suunnitteluprosessi, joka on tarkoitettu laajojen ohjelmistosovellusten tuottamiseen. Mallissa iteratiivisuus esiintyy vaiheiden välisinä takaisinkytkentöinä, joiden tarkoitus on mahdollistaa virheiden havaitseminen ennen seuraavaan vaiheeseen siirtymistä. Tämä ehkäisee tilanteet joissa alkuvaiheen virhe havaitaan vasta projektin myöhäisessä vaiheessa, jolloin virhe on ehtinyt vaikuttamaan laajalle alueelle.

Mallia esittelevässä julkaisussa painotetaan, että toteutettava sovellus tulisi suunnitella etukäteen ennen ohjelmoinnin aloittamista. Lisäksi kaikissa mallin vaiheissa tuotetaan kattava dokumentaatio. Dokumentointia tarkennetaan maininnalla, että huono määrittelydokumentaatio asettaa haasteita sovelluksen toteuttamiselle. Ennen varsinaisen toteutusvaiheen alkamista tulisi muodostaa prototyyppisovellus testattavaksi. Prototyypin tarkoituksena on teknisten ratkaisuiden toimivuuden varmistaminen hyvissä ajoin ennen sovelluksen varsinaisen toteutusvaiheen alkamista [34].

Vesiputousmallin kuvaillaan soveltuvan erityisen hyvin laajoihin ohjelmistoprojekteihin, kuten käyttöjärjestelmien ja kääntäjien kehittämiseen [3]. Mallin mukai-



Kuva 3.1 *Automaation elinkaarin vaiheiden eteneminen, tulokset sekä vaiheisiin liittyvät laadunvarmistukselliset toimenpiteet [1].*

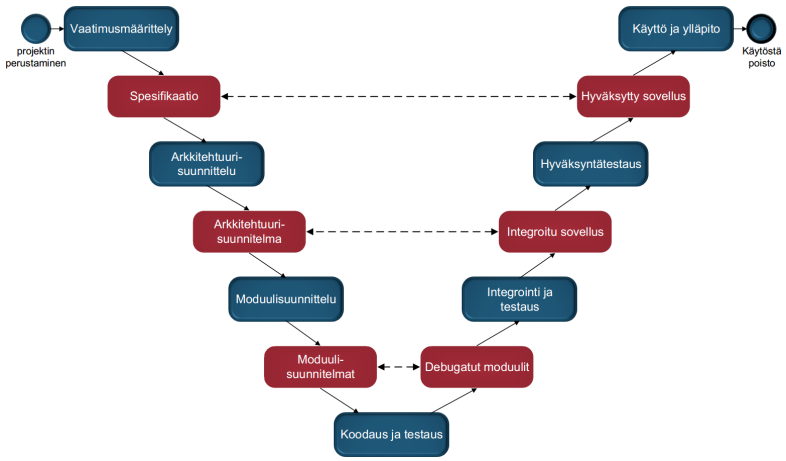


Kuva 3.2 *Vesiputousmallin mukainen sovellussuunnittelun vaihejako [34]*

sesti määrittelyvaiheen vaatimusten pohjalta valmistellaan yksityiskohtaiset suunnitelmat seuraavaa vaihetta varten. Tämä kuitenkin asettaa haasteita alun vaiheisiin tuleviin muutoksiin kesken projektin. Vesiputousmallia käytetään ohjelmistotuotannon lisäksi myös teollisuusprojekteissa. Malli soveltuu hyvin projekteihin, joissa vaatimukset ovat tiedossa ennen projektin aloitusta. Vesiputousmallissa ohjelmistokehitystä ohjataan systemaattisilla peräjälkeen suoritettavilla työprosesseilla, jotka muodostuvat usein esitutkimuksesta tai tarvekartoituksesta, sitä seuraavasta määrittely-, suunnittelu- ja toteutusvaiheista. Toteutuksen jälkeen ennen käyttöönottoa tuotettu sovellus testataan. Usein vesiputousmallia käytettäessä noudatetaan järjestystä, jossa ensin määritellään asiakastarpeet, suunnitellaan ja testataan tuote, joka lopuksi otetaan käyttöön. Vaiheiden välinen iteratiivisuus, joka esiintyy mallissa edelliseen vaiheeseen osoittavana nuolena, on usein unohdettu mallia esiteltäessä tai sovellettaessa käytäntöön [11].

Ohjelmistoprojektin elinkaarimalli v-malli esiteltiin vuonna 1986 *Software Engineering Journal* -lehdessä [33]. Julkaisussa käsitellään ohjelmistoprojektin hoitamista, suunnittelutyötä, laadunhallintaa, sekä ohjelmistoprojektin elinkaarta. V-malli on esitetty kuvassa 3.3. Julkaisussa korostetaan, että laatu on kaikkien projektiin osallistuvien vastuulla ja sen tulee näkyä omasta työstä, eikä laatua voida lisätä millään ulkoisella toimella. Laadunvarmistustoimilla voidaan kuitenkin havaita ongelmat ennen kuin ne tuottavat kustannuksia. Laadunhallinnan mainitaan muodostuvan todentamisesta ja kelpoistamisesta. Tähän mainitaan kuuluvan sovellussuunnittelussa testaaminen sekä erillinen laadunvarmistaminen, jossa laadunvarmistushenkilöstön tehtävänä on pitää huolta sovelluskehitykselle määriteltyjen menettelytapojen noudattamisesta.

Laadunvarmistustoimista testaus katsotaan v-mallissa olevan niin sanotusti sisäänrakennettu, kun se kulkee käsi kädessä eri vaiheiden edetessä. Tämä eroaa vesiputousmallista, jossa testaus sijoitettiin tapahtuvan vasta prosessimallin lopuksi. V-mallisessa lähestymistavassa jokaista vaihetta seuraa vaiheen aikana tuotetun sisällön osalta todentaminen ja kelpoistaminen. Todentamisen ja kelpoistamisen mer-



Kuva 3.3 *Alkuperäisessä muodossaan oleva v-malli [33]*

kitys määritellään alkuperäisessä julkaisussa seuraavasti :

> Verification: To establish the correspondence between a software product (documentation or code) and its specification ? **'Are we building the product right?'**
> Validation: To establish the fitness of a software product for its operational mission ? **'Are we building the right product?'** [33]

Muutostenhallinnalla (osa konfiguraationhallintaa) suositellaan pitämään kirjaa asiakkaalle toimitetun sovelluksen kokonaisuudesta. Kirjanpito sisältää sovelluksen komponentit ja konfiguraatiot, sekä sovellusta käsittelevän dokumentaation ja edellä mainittujen versiotiedot. Muutostenhallinta vastaa projektin aikana tapahtuvista muutoksista, joille luo painetta muun muassa havaitut tekniset ongelmat tai liiketoiminnan muutoksista johtuvat vaatimusmuutokset. V-mallille alkunsa antaneen vuoden 1986 julkaisun johtopäätöksissä ennustetaan, että ohjelmistotekniikan jatkuvasta kehityksestä huolimatta ohjelmistotuotannon hallinta tulee tapahtumaan esitettyjen periaatteiden mukaan vielä tulevaisuudessakin [33].

3.3 Inkrementaaliset elinkaarimallit

Edellisessä kappaleessa esitellyn vesiputousmallin ongelmina oli sen sopivuus ohjelmistotuotannon sovellusten kehittämiseen. Kritiikki kohdistetaan mallin mukaiseen tarpeettoman suureen panostukseen ja yksityiskohtaiseen määrittelydokumentaation muodostamiseen ennen sovelluksen esittelyversion toteutusta. Muun muassa tähän asiaan puututaan evoluutiomallin avulla, jossa sovellus toteutetaan muodostamalla ensin toimiva kokonaisuus pienellä määrällä toiminnallisuuksia. Tämä suoritetaan aikaisessa vaiheessa saatavan asiakaspalautteen vuoksi. Toiminnallisuuksia lisä-

tään myöhempiin kehitysversioihin. Näin asiakaspalaute saadaan mukaan projektin aikaisessa vaiheessa. Evoluutiomalli kuitenkin muistuttaa yksinkertaisinta ohjelmistoprojektimallia, jossa sovellusta kehitetään ainoastaan koodia lisäämällä ja muokkaamalla. Projektin kehityskulun suunnittelemattomuuden takia uusien lisäysten seurauksena koodin rakenne muodostuu samalla tavoin jäsentelemättömäksi. Mikäli koodin rakenteen hallitsemiseksi vesiputousmallin vaihejakoa hyödynnetään evoluutiomallin iteraatioissa, tuo se mukanaan vesiputousmallin ongelmia. Lisäksi alun esittelyversioon tilapäisiksi tarkoitetut ratkaisut muodostuvat perustaviksi päätöksiksi joita voi olla myöhemmin mahdoton muuttaa [3].

Spiraalimalli on esitelty vuonna 1986 julkaisussa *A Spiral Model of Software Development and Enhancement* [3]. Iteratiivinen spiraalimainen kehitysmalli painottuu riskienhallintaan. Sovelluskehitys tapahtuu peräkkäisinä iteratiivisina spiraalin kierroksina. Ensimmäisessä spiraalin neljänneksessä määritellään tavoitteet, ratkaisuvaihtoehdot ja rajoitteet. Esimerkiksi kaksinkertaistaa tuottavuus viidessä vuodessa. Tavoitteen saavuttamiseksi ratkaisuvaihtoehtona on työkäytäntöjen muutos asetetuilla kustannusrajoituksilla. Seuraavan neljänneksen vaiheessa arvioidaan ratkaisuvaihtoehtoja. Ensin arvioidaan riskit, sitten panostetaan paikannettuihin epävarmoihin ja riskialttiisiin kohtiin. Kokenut kehittäjä osaa keskittyä selvittämään epävarmoja kohtia mahdollisimman aikaisessa vaiheessa. Kolmannen vaiheen sisällön määrää projektissa jäljellä olevat riskit. Viimeisessä neljänneksessä arvioidaan resurssit ja tehdään toimintasuunnitelma seuraavaa kierrosta varten [28].

Spiraalimallin rakenteellinen muoto määritellään riskianalyysin avulla. Jos esimerkiksi projektissa on matala riski asiakasvaatimusten muuttumiseen mutta tiukka aikataulu tai budjetti, spiraalimalli muotoutuu vesiputousmalli -tyyppiseksi. Tällöin sovellus toteutetaan yhdellä vesiputousmallin vaiheet sisältävällä suunnittelukierroksella, jolloin ensimmäinen prototyyppi on valmis sovellus. Tällöin spiraalimallin riskienvähentämismenettelyt tiedostetaan, mutta niitä ei suoriteta. Toisessa tapauksessa jos projektissa on korkea riski asiakasvaatimusten muuttumiseen, mutta matala riski aikataulun, budjetin ja teknisen toteutuksen suhteen, spiraalimallista ohjelmistoprojektia voidaan toteuttaa evoluutiomallia mukaillen. Tällöin spiraalimallin mukaisten spesifikaatioiden laatiminen jätetään suorittamatta [3].

Ohjelmistoprojektin elinkaarimalli *Rational Unified Process* (RUP) on yhdistelmä vesiputousmallia ja spiraalimallia. UML-mallinnuskielellä on mahdollista spesifioida ja dokumentoida oliopohjaisen sovelluksen suunnittelu. RUP-mallissa vaatimukset kerätään UML:n esittelemiksi käyttötapauksiksi, jotka kuvaavat toiminnalliset vaatimukset käyttäjän ja sovellukseen liitettävien järjestelmien näkökulmasta.

Käyttötapauksia hyödynnetään myös jäljitettävyyden toteutumiseen. Ne yhdistävät vaatimukset toteutetun sovelluksen osiin. Kehitys jaetaan käyttötapauksien avulla pienempiin, kerrallaan toteutettavissa oleviin iteraatioihin. Jokainen näistä

sovelluskehityksen iteraatioista voi itsessään sisältää vaatimusmäärittelyn, suunnittelun ja toteutuksen sekä testauksen. Sovelluksen testauksen suunnitteluun hyödynnetään edellä mainittuja käyttötapauksia muodostamalla jokaisesta käyttötapauksesta oma testitapauksensa. Näin voidaan todentaa sovelluksen toteuttavan määritellyt toiminnallisuudet testaussuunnitelman esittämällä tavalla. Testitapaukset dokumentoidaan tallentamalla testiraportit [28].

3.4 Ketterät menetelmät

Kevyiden ohjelmistotuotannon menetelmien suosijat kehittivät raskaiden kehitysprosessien vastapainoksi ketteräksi kutsutun kehitystavan. Vuonna 2001 perustettu ketterien menetelmien edistämistä ajavan järjestön julkaisu *Agile Manifesto* määrittelee, että sovelluskehityksen tukemisen (prosessit, työkalut, dokumentaatio) toiminnot ovat vähemmän tärkeitä. Tärkeintä on tyytyväinen asiakas ja toimiva ohjelmisto [11, s. 44]. Verrattuna muihin ohjelmistomalleihin ketterän kehitysmallin mainitaan paremmin vastaavan projektimuutosten tuomiin haasteisiin. Ketterän mallin mukaisesti kehitettävä sovellus etenee nopeina pyrähdyksinä. Projektin muuttuneet vaatimukset otetaan mukaan käsiteltäväksi vasta seuraavan pyrähdyksen aikana. Näin kehitettävästä sovelluksesta tehdään uusia versioita lyhyiden, noin kahden viikon mittaisten pyrähdysten aikana. Ketterien menetelmien etuihin luetellaan myös suunnitteluryhmän parantunut keskinäinen tiedonvaihto ja nopea asiakaspalautteen huomioiminen. Kuitenkaan laajempiin ohjelmistohankkeisiin ja koneenohjauksiin ketterää kehitysmallia on vaikea tai mahdoton soveltaa [11, s. 45]. Reaaliaikajärjestelmien kehittämiseen ketterää toimintatapaa edustavaa Harmony/ROPES, joka yhdistää ketterään menetelmään raskaita perinteisiä toimintatapoja [21].

Ketterän kehitysmallin pyrähdykset suunnitellaan ennen iteraatiokierroksen aloittamista. Tarkoituksena on valita pyrähdykseen suoritettavaksi tärkeimpiä tehtäviä siten, että pyrähdyksen ajallinen kesto pysyy vakiopituisena ja lyhyenä. Päivittäin pidettävässä palaverissa käydään läpi edellisen päivän edistyminen ja ilmenneet ongelmat sekä alkavalle päivälle suunnitellut toteutettavat tehtävät. Iteraatiopyrähdyksen lopuksi sovellus esitellään asiakkaalle pikaisen palautteen saamiseksi. Palautteen avulla johdetaan uusia vaatimuksia seuraavia pyrähdyksiä varten. Suunnittelutiimi pitää lopuksi takautuvan tarkastelun siitä, mikä pyrähdyksen aikana meni hyvin tai huonosti, jotta toimintaa voidaan kehittää seuraavaa iteraatiopyrähdystä varten [12].

Ketterän kehityksen menetelmiä ei puhtaasti yritetä soveltaa käytännössä, vaan sen ideoita hyödynnetään osana ohjelmistoprojekteja. Yleisin käytössä oleva ketterään kehitysmalliin pohjaava projektimalli Scrum on tapa organisoida spiraalimaisen kehitysprosessin iteraatiot. Se ei ole varsinainen ohjelmistoprojektin hallintamenetelmä, sillä se ei ota kantaa käytettyihin kehitysmenetelmiin eikä työn organisointiin

[11, s. 54].

Ketterän kehitysmenetelmän Scrumin liittyviksi hyödyiksi ohjelmistoteollisuuden yrityksissä on mainittu menetelmän joustavuudesta aihetuvan työtyytyväisyyttä ja motivaation kohentumista. Ketterä kehitysmalli luo positiivisia vaikutteita työympäristöön yhteisen päätöksenteon, palautteen sekä sosiaalisuuden lisääntyminen päivittäisesti pidettävien palaverien myötä. Asiakastyytyväisyyteen vaikuttaa avoimempi kommunikaatio sekä asiakkaan läheisempi osallistuminen kehitysprosessiin. Ongelmiksi lukeutuu Scrumin sovittaminen yrityksen vanhoihin toimintatapoihin. Myös pyrähdyksen aikainen työ voi muodostua hankalaksi rauhoittaa siihen liittyvälle työlle johtuen vika- ja ylläpitotöistä. Lisäksi pyrähdyksen aikaista työtä joutuu usein suojelemaan muutosyrityksiltä kesken pyrähdyksen [51].

Muita ongelmia inkrementaalisten ja iteratiivisten elinkaarimallien mukanaan tuoma ketterä suunnittelutiimi voisi soveltua automaatiosovellusten kehittämiseen, mutta pääosa kohdeyrityksen automaation sovellusprojekteista on kooltaan merkittävästi suppeampia kuin ohjelmistoteollisuudessa. Tällöin ketterien menetelmien vaatimat resurssit olisivat ylimitoitettuja. Koska paljon resursseja vaativia suuria projekteja suoritetaan kohdeyrityksessä kuitenkin harvemmin, unohtuisi ketterä toimintatapa työntekijöiltä ajan saatossa ennen seuraavan laajan projektin alkua.

Muita ongelmia alkuperäisessä vesiputousmallissa mainitaan muodostuvan ohjelmistotuotteen testaamisen sijoittuminen koko toteutuksen jälkeen [3]. Vasta tällöin havaitut virheet voivat ehtiä vaikuttamaan jo laajalle alueelle sovellusta. Vesiputousmallia hyödyntävät yritykset ovat jalostaneet toimintatapojansa ottamalla elinkaarimallin rinnalle vesiputousmallin [34] ulkopuolisia projektin tukitoimintoja, joita käsiteltiin luvussa 5. Tämän seurauksena tänä päivänä vesiputousmallista on liikkeellä useita eri muunnelmia, joissa edellä mainittuihin ongelmiin on kehitetty vastaus. Vesiputousmallin eräässä esityksessä [39, s. 30] ohjelmistosovelluksen toteutusvaihe jaotellaan pienempiin osiin ja testaus sijoitetaan tapahtuvaksi heti näiden jälkeen. Kotimaisessa kirjallisuudessa [1],[12] tästä käytetään termiä laadun sisään rakentaminen v-mallin mukaisen elinkaarimallin yhteydessä.

4. LAADUNHALLINNAN PERIAATTEET

Laatu nähdään usein tarkastelijasta riippuvana subjektiivisena käsitteenä, jolloin laadulle ei voida asettaa yksiselitteistä mittaa. Laatu merkitsee tuotteen tai palvelun kykyä tyydyttää asiakkaan suoraan tai välillisesti ilmaistut tarpeet [1, s. 5]. Ohjelmistosovelluksen laatu voidaan jakaa osatekijöistään muodostuviin komponentteihin. Laatu muodostuu tällöin subjektiivisen osan lisäksi myös objektiivisesta komponentista, joka on arvioitavissa. Näiden kahden lisäksi ohjelmistototeutuksissa vaikuttaa myös kolmas komponentti, joka muodostuu arvioimattomissa olevista tekijöistä [12].

Laadun objektiivinen komponentti sisältää arvioitavissa olevat asiat, eli kuinka hyvin tuote vastaa vaatimusmäärittelyjä. Subjektiivinen laatukomponentti koostuu asiakkaan näkemyksistä, kuten kuinka hyvin tuote täyttää asetetut vaatimukset ja kuinka helppokäyttöinen tuote on. Arvioimattomissa olevat tekijät -komponenttiin kuuluu muun muassa miten tuote skaalautuu asiakkaan tuleviin ja vielä tuntemattomiin tarpeisiin tai miten tuote käyttäytyy ennalta odottamattomissa virhetilanteissa. Tämän laatukomponentin vaatimuksiin vastataan valitsemalla suunnitteluratkaisut asianmukaisesti [12].

Ohjelmistojen ja sovellusten luotettavuuden varmistamiseksi yrityksissä hyödynnetään laadunhallintajärjestelmää. Automaatioalalla taustalla on yleisen laatuajattelun lisäksi ohjelmoitavien järjestelmien hyödyntäminen turvallisuuskriittisissä kohteissa, joihin liittyy vakavia riskitekijöitä. Sovelluksen laatuun voidaan vaikuttaa erityisesti systemaattisella suunnitteluprosessilla sekä erillisillä laadunvarmistustoimilla. Tässä luvussa esitellään yleisellä tasolla organisaation laadunvarmistusta tukevat tekijät, sekä tekijät jotka vaikuttavat ohjelmistosovelluksen laatuun. Lisäksi esitellään sovelluksia vaivaavia laatupoikkeamia, joita laadunvarmistuksen menetelmillä pyritään havaitsemaan.

4.1 Laadunhallintajärjestelmä

Kansainvälinen standardi ISO 9001 [37] määrittelee laadunhallintajärjestelmää koskevat vaatimukset. Vaatimukset ovat yleisiä ja niitä voidaan soveltaa kaikille organisaatioille tuotetuista tuotteista riippumatta. Standardin vaatimuksia voidaan hyödyntää organisaatioissa, jotka haluavat näyttää kykynsä toimittaa asiakasvaatimukset täyttäviä tuotteita, tai jos organisaatio pyrkii soveltamaan järjestelmää

joka sisältää jatkuvan parantamisen, asiakasvaatimusten, lakien sekä viranomaisvaatimusten täyttymisen varmistavat prosessit. [37] Asiakkaalla voi olla tarve osoittaa automaatiosovelluksen laatu viranomaisille tai loppuasiakkaalle. Osoittaminen vaatii dokumentoitua laadun suunnittelua ja laadunseurantaa sekä todistusaineiston kokoamista koko projektin ajan. Tämä vaatii automaatiotoimittajalta totuttua laajempaa dokumentointia [1, s.8].

Organisaation toimintamalleissa käytettyjen resurssien avulla saadaan aikaan tuotoksia. Prosessimainen toimintamalli muodostuu prosessijärjestelmän soveltamisesta organisaatiossa. Organisaation tulee määrittää prosessit ja johtaa niitä siten, että ne tuottavat toivotut tulokset. ISO 9001 standardin mukaan prosessimainen toimintamalli mahdollistaa asiakastyytyväisyyden parantamisen erityisesti silloin, jos käytössä on hyvä laadunhallintajärjestelmä [37].

ISO 9001 vaatimusten mukaan organisaation tulee luoda, dokumentoida ja toteuttaa laadunhallintajärjestelmä jota ylläpidetään ja jonka vaikuttavuutta parannetaan jatkuvasti. Standardin mukaan organisaation tulee määrittää laadunhallintajärjestelmää varten tarvittavat prosessit sekä seurata, mitata ja analysoida näitä prosesseja. Lisäksi ISO standardin mukaan organisaation tulee määrittää ja varata resurssit laadunhallintajärjestelmän toteuttamiseen ja ylläpitämiseen sekä jatkuvaan parantamiseen [12].

Laadunhallintajärjestelmän vaatimustenmukaisuus varmistetaan sisäisten auditointien avulla. Auditoinnit suoritetaan suunnitelluin aikavälein ja niiden avulla määritetään onko laadunhallinta vaikuttavasti toteutettu ja ylläpidetty. Auditointiohjelmassa määritellään kriteerit, laajuus, suoritustaajuus, sekä menettelyt. Oman työn auditointi ei ole tarkoituksenmukaista eikä sallittua. Siksi auditoijat valitaan niin, että auditointiprosessin objektiivisuus toteutuu. Organisaation menettelyohjeessa määritellään auditointien suunnitteluun, suorittamiseen, tallenteiden luomiseen ja tulosten raportoimiseen liittyvät vastuut ja vaatimukset [37].

Organisaation arviointitilaisuudessa tutkitaan täyttääkö yrityksen laadunhallintajärjestelmä standardin vaatimukset. Yrityksen laatusertifikaatin hakeminen aloitetaan laatujärjestelmän kehitystyöllä, jossa kartoitetaan yrityksen nykytilanne ja asetetaan tavoitteet. Laatujärjestelmän arvioinnin suorittaa sertifioinnin myöntävä taho. Arviointi tapahtuu dokumentoidun materiaalin sekä haastattelujen avulla kerätyn aineiston tutkimisella. Sertifikaattia voidaan hakea koko yritykselle tai vain osalle yritystä ja sen voimassaolo vaatii väliajoin suoritettavia uusinta-arviointeja. Sertifikaatti voi muodostua yrityksellä kaupansaannin edellytykseksi. Myös sertifioidun yrityksen asiakas voi ottaa yhteyttä sertifikaatin myöntäneeseen organisaatioon, mikäli huomaa että yritys ei toimi väittämällään tavalla [12].

4.2 Laadun mittaaminen

Edellisessä luvussa käsiteltiin, kuinka yrityksen laadunhallinnan mukainen laatupolitiikka määrittelee toiminnan tavoitteet sekä yritykselle tärkeät asiat. Yrityksen johdon tehtävänä on määritellä itselleen toimintatavat ja pyrkiä optimoimaan toimintaansa laatupolitiikan mukaisilla laadunkehityshankkeilla. Näistä laatutavoitteista muodostuu laadunhallintajärjestelmä, joka esiintyy joko dokumentoituina tai dokumentoimattomina työprosesseina joilla tuote yrityksessä syntyy. Yrityksen laatujärjestelmää kehitettäessä mittaaminen on keskeinen tekijä, joka kohdistetaan niin tuotantoprosessiin kuin tuotteeseen [12].

Laadun mittarit koostuvat kahden tyyppisistä mittareista. Strategiset mittarit vastaavat pitkän aikavälin tapahtumista liiketoimintanäkökulmasta. Operatiivisilla tarkastellaan päivittäin kerättävää ja päätöksen tekoa tukevaa tietoa. Operatiivisilla mittareilla saadaan tietoa projektin kulusta ja siitä, miten tavoitteet on saavutettu. Operatiivisen mittarin kohteena voi olla tuotteen mittaaminen, tai tietoa voidaan kerätä työprosessista jolla tuotetta tehdään. Mittaamisen ongelmiksi luetellaan kustannukset, työntekijöille muodostuvat asenneongelmat mittareiden käyttöönottamisesta, sekä se seikka ettei yksittäisiä luotettavia mittareita ole olemassa [41]. Yrityksen työntekijöillä pitää olla tiedossa että laadun mittaukseen käytettävää tietoa ei käytetä työntekijöiden työtehon keskenään vertailuun, vaan tarkoituksena on kehittää työprosessia [37].

Hyvien mittareiden löytymistä vaikeuttaa työn luonteen ja määrän muuttuminen eri projektien tai tuotantokertojen kesken. Mittareiden tulee olla mahdollisimman automaattisia sekä niitä tarvitaan oikea määrä, jotta analysoitavaa informaatiota saadaan tarpeeksi ja trendin suunta voidaan havaita luotettavasti. Mittareita käyttöönotettaessa kiinnitetään huomiota mittaustulosten käyttöön ja siihen, että käyttö olisi kaikilla työntekijöillä selvä, eikä mittaustuloksia käytetä väärin. Väärinkäytöltä vältytään, kun mitataan usean henkilön työryhmää tai prosessia eikä yksilöä. Työprosessia mitattaessa kohteiksi valikoituvia mittauksia ovat arvioitujen aikataulujen vertaaminen toteutuneisiin, työprosessin tuottavuus, laskutuskelvottomien tuntien muodostuminen, katselmoinneissa löytyneet virheet ja virhekorjausten kustannukset. Tuotteen laadun mittaamiseen liittyvä asiakastyytyväisyys mitataan asiakastyytyväisyyskyselyllä, joka suoritetaan toimituksen jälkeen [41].

Ilman mittareita on hankala tietää kuinka laatua tulisi kehittää tai kuinka laadunkehityshanke etenee. Tiedettäessä yrityksen laatujärjestelmä ja kehityshanke, valitaan sopiva mittarijoukko joilla pystytään kartoittamaan havaittuja ongelmakohtia. Seuranta- ja mittausprosesseja käytetään varmistamaan tuotteen vaatimustenmukaisuus. Näillä menetelmillä osoitetaan myös laadunhallintajärjestelmän vaatimustenmukaisuus sekä parannetaan sen vaikuttavuutta. Myös prosessien kykyä

saavuttaa suunnitellut tulokset seurataan sopivien menetelmien avulla. Näitä määritettäessä organisaatiossa huomioidaan minkä tyyppinen ja laajuinen seuranta tai mittaus kullekin prosessille on tarkoituksenmukaista suhteessa prosessin merkitykseen laadunhallintajärjestelmän vaikuttavuuden osalta [37].

4.3 Toteutettavan tuotteen laadunhallinta

Standardissa ISO 9001 yksi laadunhallintajärjestelmän vaatimuksista on tuotteen toteuttamisen suunnittelu. Suunnittelu toteutetaan organisaation toimintatapoihin soveltuvassa muodossa. Tässä kappaleessa käsitellään standardissa kuvatut asiat tuotteen toteuttamisen ja laadunvarmistamisen kannalta. ISO 9001 sertifioidussa organisaatiossa tulee määritellä prosessit joita hyödynnetään tuotteen toteuttamiseen. Suunnitteluun tulee sisällyttää tuotteen laatuvaatimukset, tuotekohtaiset tarpeet vaadittavien prosessien ja resurssien suhteen sekä tuotekohtaiset todentamisen ja kelpoistamisen toimenpiteet hyväksymiskriteereineen [37]. Tässä esitellään laadunvarmistuksen muodostavat todentamisen ja kelpoistamisen toimenpiteet yleisellä tasolla sekä yksityiskohtaisemmin sovelluksen todentamisen osalta.

Toteutettavaa tuotetta varten organisaatio määrittää vaatimukset joita tuotteen aiottu käyttötarkoitus edellyttää. Asiakasvaatimukset määritetään kirjallisesti. Määritellyissä vaatimuksissa huomioidaan tuotetta koskevat lait ja viranomaismääräykset sekä toimituksen jälkeiset vaatimukset. Ennen kuin organisaatio sitoutuu toimittamaan tuotteen asiakkaalle, katselmuksella varmistetaan että tuotevaatimukset tulevat määritellyiksi ja että organisaatio kykenee täyttämään asetetut vaatimukset [37].

Toteutettavan tuotteen kehityksen ja suunnittelun ohjaaminen toteutetaan määrittelemällä suunnittelun ja kehittämisen eri vaiheet sekä näihin liittyvä todentaminen ja kelpoistaminen. Organisaatiossa kiinnitetään huomioita myös suunnitteluun ja kehitykseen osallistuvien vuorovaikutukseen, jotta viestintä ja vastuut ovat selkeästi määriteltyinä. Tuotteen vaatimusten lähtötiedoiksi ISO standardin mukaan kuuluu toiminnalliset vaatimukset sekä ei-toiminnalliset suorituskyky vaatimukset, lakien ja viranomaisten vaatimukset, vastaavanlaisista aikaisemmista suunnitelmista kerätty informaatio sekä muut suunnittelun ja kehittämisen kannalta oleelliset asiat. Lisäksi lähtötietojen asianmukaisuus tulee katselmoida ja niissä esitettyjen vaatimusten tulee olla kattavia ja yksiselitteisiä eivätkä ne saa olla ristiriidassa keskenään [37]. Edellä mainitut asiat on poimittu standardista koska ne koskevat sovellussuunnittelua.

Todentaminen ja kelpoistaminen suoritetaan määritellyn prosessin mukaisesti, jotta varmistutaan että tuote täyttää lähtötietojen vaatimukset. Todentamisen ja kelpoistamisen tuloksista ja toimenpiteistä pidetään tallenteita. Nämä muodostavat todisteet tuotteen laatutasosta. Kelpoistaminen tehdään ennen tuotteen toimitta-

mista tai käyttöönottoa mikäli se käytännössä on mahdollista [37].

4.4 Laadunvarmistuksen ongelmat

Tässä luvussa esitellään yleisiä sovellussuunnittelun laadunvarmistukseen liittyviä ongelmia. Kyseessä on ohjelmistoteollisuudesta raportoituja tapauksia, jotka osaltaan soveltuvat tämän työn kontekstiin. Lisäksi mukana on joitain automaatioprojekteista havaittuja asioita. Usein laadunvarmistuksen näkökulmat huomioidaan puutteellisesti ohjelmistotuotantoalalla sovellussuunnittelun parissa. Tällöin ainoastaan testaaminen huomioidaan, mutta muita laadunvarmistustoimenpiteitä sekä laatukriteereitä pohditaan vasta ongelmien ilmaantuessa, jos silloinkaan. Mikäli laatukriteerejä ei ole asetettu, ei laadunvarmistustoimiakaan voida kohdentaa oikein [22, s. 88]. Lisäksi laadunvarmistuksen toimenpiteitä jätetään suorittamatta projektikiireiden takia. Tähän voi vaikuttaa projektisuunnitelman kiireellinen aikataulu tai suunnittelijan puutteellinen oman työn suunnittelu.

Laadunvarmistusprosessin määrittelyn puuttumisesta seuraa laadunvarmistuksen sivuuttaminen kokonaan. Kehitettävän tuotteen tai vaihetuotteen oikeellisuutta ei pystytä todentamaan jos menetelmissä on puutteita. Yleisellä tasolla kuvatun prosessin lisäksi tulisi määritellä laadunvarmistustoimien yksityiskohtaiset mekanismit. Dokumentoinnin heikko laatu esiintyy esimerkiksi vaadittavien testausraporttien, ohjelmakoodin kommentoinnin tai ylläpitovaiheessa tarvittavien dokumenttien puuttumisena. Tähän vaikuttaa se, ettei dokumentteja ole totuttu tekemään koska niitä ei projektinjohdon toimesta vaadita. Tämän seurauksena ei ole olemassa valmiita dokumenttipohjia, joiden avulla dokumentointi selkiytyisi [22]. Automaatiojärjestelmää kuvaavien dokumenttien puutteet vaikeuttavat ylläpitotyötä ja laadunvarmistuksen osalta toimitetun järjestelmän kelpoistaminen ja asiakkaalta saatava hyväksyntä hankaloituu.

Tiedonvälityksen ongelmat näkyvät usein vasta projektin loppupuolella. Tällöin käyttöönoton yhteydessä voidaan havaita, että kohde ei vastaa enää jäädytettyjä lähtötietoja jonka mukaan sovellus on suunniteltu tai että kaikkia käyttöönoton aikana tulleita muutoksia ei ole dokumentoitu mitenkään. Ensin mainitussa tapauksessa syy ei useinkaan löydy suunnittelijan aktiivisuuden puutteesta, vaan kyseessä on todennäköisesti projektin viimehetken muutoksiin liittyvät tiedonvaihdon puutteet. Ongelmia voi muodostua myös projektin sisäisen tiedonvaihdon myötä, jos kaikkia esitettyjä asiakasvaatimuksia ei ole dokumentoitu tai myydyt ominaisuudet eivät välity suunnittelijalle. Käyttöönoton aikaisten muutosten yhteydessä tapahtuu helposti muutostenhallinnan ohittaminen. Tällaisessa tilanteessa sovellussuunnittelija on käyttöönottamassa järjestelmää loppuasiakkaan luona, kun käyttöönoton yhteydessä nousee esiin muutostarpeita. Nämä toteutetaan suoraan koodiin ja muutos ei kulje tarvittavan tiedonvälitysketjun läpi, eikä täten noudata muutostenhallinnan

menettelyä. Tästä seurauksena dokumentaatio jää päivittämättä, jolloin tehdystä muutoksesta ei jää mitään jälkeä dokumentaatioon.

Testaaminen on jää usein sovellussuunnittelussa ainoaksi laadunvarmistustoimeksi. Sen vaatima resursointi osataan yleisesti sovellussuunnittelutyössä huomioida projektisuunnitelmissa. Ongelmia aiheutuu testaamisen suunnittelemattomuus. Testausvaihetta varten vaadittavia laatukriteerejä ei useinkaan ole kirjattu. Ohjelmistoteollisuudessa moduuli- ja integrointitestausta ei välttämättä tällöin suoriteta ja testaaminen siirtyy suoraan järjestelmätestaukseen. Testaukseen liittyviä ongelmia aiheutuu myös testiympäristön riittämättömyydestä yhtäaikaiseen testaamis- ja kehitystyöhön. Testaamisympäristön parissa työskentelevien työntekijöiden koordinointia ei myöskään välttämättä huomata hallita vaaditulla tavalla työn tuottavuuden kannalta [22].

Laatukriteerien arviointia vaikeuttaa testaustausympäristön merkittävä poikkeama varsinaisesta järjestelmästä jossa kehitettävää sovellusta tullaan suorittamaan. Myös tietoturva muodostaa erään laatukriteerin, jonka verifiointiin ei juurikaan käytetä ohjelmistotuotannossa resursseja. Sitä ei useinkaan huomioida projektisuunnitelmissa, eikä sen testaamista suoriteta asiantuntemuksen puuttumisen ja tästä seurauksena olevan testausmenettelyjen puuttumisen takia. Tähän vaikuttaa myös testausympäristöjen poikkeavuus lopullisesta suoritusympäristöstä, jolloin varsinaisia tietoturvauhkia ei voida välttämättä testata tai arvioida [22].

Loppuasiakkaalta myöhässä saatava palaute aiheuttaa ongelmia, kun sitä ei voida enää normaalin projektin puitteissa huomioida ja se siirtyy lisätyöksi. Tällöin muutosten toteuttaminen hankaloituu, esimerkiksi tuotannossa olevalla laitoksella ajossa olevan sovelluksen muuttamiseen liittyvien haasteiden myötä. Automaation osalta tähän ongelmaan vastataan tehdastestauksella, johon asiakkaan aktiivinen osallistumisen myötä esiin tulleet muutostoiveet voidaan huomioida projektin aikana ja asiakkaan näkökulmasta ilman lisäkustannuksia.

Lopullisen hyväksyntätauksen ongelmat muodostuvat hyväksymissuunnitelman uupumisesta. Tätä edesauttavat epätarkat määrittelyt, puutteellinen aikataulu tai aikataulusta johtuva kiire sekä asiakkaan mielenkiinnon puute. Kiireestä johtuen testit jäävät lyhyiksi tai testaaminen lopetetaan kesken, jolloin tarvittavan kattavaa testausta eri suoriteta. Testitulosten luotettavuudesta ei voida tällöin varmistua ja ongelmia siirtyy tuotantoympäristöön, jonne toteutettavat muutokset voivat usein olla ongelmallisia [22].

4.5 Katsaus aikaisemmista lähestymistavoista

Laadunhallintajärjestelmän kehityshankkeessa [49] pitkän aikavälin kehityksen tavoitteena on ohjelmistotuotannon saattaminen ISO 9001 standardin mukaiselle tasolle. Yhtenä osana hankkeen edistämistä muodostettiin projektinhallintaa varten

projektisuunnitelmarunko, jonka pohjalta tulevia projekteja suunnitellaan ja toteutetaan. Tämän lisäksi käsiteltävänä oli dokumentoinnin yhdenmukaistaminen, tarkastuksien avulla virheiden siirtymisen seuraavaan vaiheeseen estäminen, testauksen avulla sovelluksen spesifikaation mukaisuudesta varmistuminen sekä tuotteenhallinnan ottaminen mukaan sovellusprojekteihin. Kehityshankkeen tuloksia siirretään laadunhallintajärjestelmän dokumentaatioon tarkoituksena rakentaa sitä lähemmäs ISO standardin asettamia vaatimuksia.

Laadunvarmistusprosessin kehittämistä käsittelevässä Hanna Leppälän diplomityössä *Laadunvarmistusprosessin kehittäminen kaupankäyntiohjelmistojen kehityshankkeissa* [22] käsittelee ohjelmistotuonnosta vastaavaa tilannetta kuin tämä työ. Siinä asiantuntijahaastattelujen ja kokemuksen perusteella havaittuihin ongelmiin valitaan alan kirjallisuudesta korjaavia toimintoja jotka kootaan kehitysehdotukseksi. Laadunvarmistamistamisen kehittämiseksi suositellaan yhdenmukaisien laatukriteerien laatimista, laadunvarmistustoimien ohjeistusta ja tarkastusmenetelmien käyttöönottoa.

Ohjelmistoprojektien laadullisia haasteita [49] käsittelevässä opinnäytetyössä havaittiin ongelmia esiintyvän, koska kehitettävien sovellusten laadullisia tavoitteita ei ollut selkeästi asetettu ja laadullisia määrittelyjä ei ollut kuvattu vaadittavalla tarkkuudella. Suosituksina havaittuihin ongelmiin ehdotetaan suurien projektien pilkkomista pienempiin osiin, vaatimusten jäljitettävyyden parantamista sekä tiiminvetäjäksi valitulta henkilöltä edellytyksiä toimia tehtävässään. Kokemusperäisistä syistä CMMI-kypsyysmallin hyödyntämistä kuvaillaan liian raskaaksi ja ehdotetaan laatukäytäntöjen kehittämistä dokumentoimalla yrityksessä käytettäviä sisäisiä prosesseja ennen kypsyysmallin mukaista kehitystä.

Edellä mainituissa aikaisemmin toteutetuissa laatuun liittyvissä kehitystä käsittelevissä töissä sovellussuunnittelun laadunvarmistusprosessin kehittämisellä on pyritty vähentämään laatupuutteiden seurauksena nousevia kustannuksia, lisätä tuottavuutta sekä parantaa projektikokonaisuuden ohjausta [42]. Prosessien kehittämiseen on etsitty menetelmiä ja malleja ohjelmistoteollisuuden kirjallisuudesta [22]. Kehitysprosessi on aloitettu kartoittamalla yrityksen nykyinen tila [49],[42],[22]. Tämän perusteella voidaan tunnistaa sovellussuunnittelun sekä laadunvarmistuksen prosesseista kirjallisuudessa esitettyjä tekijöitä. Tähän on hyödynnetty asiantuntijahaastatteluja [49] ja tämän lisäksi yrityksessä kerättyä kokemusta [22]. Asiantuntijahaastattelujen myötä on voitu selvittää työprosesseissa esiintyviä ongelmakohtia [22],[49]. Havaituista puitteista ja ongelmista on lähdetty kehittämään toimintaa laadunvarmistuksen hallinnan osalta, tarkastusmenettelyjä käyttöönottamalla, dokumentointia kohentamalla, asiakkaan ja toimittajan välistä kommunikaatiota parantamalla sekä testauksen eri osa-alueita kehittämällä [42],[22],[49].

4.6 CMMI kypsyysmallin mukainen lähestymistapa

Capability Maturity Model Integration (CMMI) on Software Engineering Instituten ylläpitämä viitekehys, joka mahdollistaa sovellussuunnittelua liiketoimintanaan suorittavan organisaation prosessien kehittämisen. CMMI:n mukaisessa kehityksessä laatuasiantuntijoiden toimesta yrityksen tuottaman tuotteen laatu liitetään työprosessin kypsyyteen jolla tuote toteutetaan [28, s. 43]. Seuraavaksi tutustutaan lyhyesti CMMI:n tarjoamaan sovellussuunnitteluorganisaatiota käsittelevään kypsyyden arviointiin ja kehittämiseen. Tässä esitettävät asiat ovat lähtöisin julkaisusta *CMMI for Development, version 1.3* (2010) [43].

CMMI toimii välineenä jonka avulla yrityksen sovellussuunnitteluprosessia tarkastellaan ja kehitetään. Tämän työn osalta keskitytään ainoastaan tarkastelemaan sovellussuunnittelun laadunvarmistuksellisia tekijöitä, koska projektin johdolliset asiat asettuvat tämän työn alueen ulkopuolelle. Kuitenkin on huomioitavaa että CMMI:n implementoiminen organisaation kypsyyden kehittämiseen vaatisi kaikkien siinä kuvattujen osa-alueiden tarkastelemista tasapuolisesti.

CMMI:n hyödyntäminen on projekti siinä missä muutkin projektit, joten se tarvitsee projektin johtamisen ja projektisuunnitelman onnistuakseen. Kehityksen saavuttaminen vaatii usein lukuisia kehityksen syklejä, joiden aikana vaikutetaan yhteen tai useampaan prosessin osa-alueeseen. Kehityksen aloittamiseksi tulee määritellä yrityksen kypsyystaso, jonka jälkeen ensimmäisenä kehitysaskeleena usein pyritään täyttämään kypsyystason 2 vaatimukset.

Mallin mukaisessa kehityksessä prosessien osa-alueita kehitetään aloitteiden avulla. Organisaation tarpeet tarkastellaan arvioinnin kautta, jonka tarkoituksena on havaita vahvuudet ja heikkoudet. Saadut tulokset analysoidaan kehityssuunnitelman formuloimiseksi. Kehityssuunnitelma pannaan käytäntöön ja kehittyminen varmistetaan auditointien avulla. Kehityksen kulun askeleet toteutetaan aina uudestaan, kunnes haluttu lopputulos on saavutettu. Tämä muodostaa CMMI-mallin mukaisen jatkuvan kehityksen käytänteen.

Kehitettävä kohde asetetaan tarkoituksenmukaisen kriittiseen tarkasteluun ja pohditaan kohdistettavia kehitystoimia. Toimiksi lukeutuvat CMMI:n tarjoamat parhaat käytänteet prosessien kehittämiseen. Koska kehittämisaloitteet eivät aina onnistu odotetulla tavalla, pyritään epäonnistumisen välttämiseksi arvioimaan ja hallitsemaan riskejä. Yleisimmiksi epäonnistumisen aiheuttaviksi syitä ovat osallistujilta uupuva sitoutuminen kehitysprojektiin, projektinjohdon määrätietoisuuden uupuminen, käyttötarkoitukseensa soveltumattomat uudet työprosessit, sekä riittämätön koulutus uusien työprosessien käyttöönottamiseen [28, s. 77].

Uusien työprosessien käyttöönottamiseen kuuluu henkilöstön kouluttaminen. Uudet työprosessit otetaan käyttöön uusissa tai juuri alkaneissa projekteissa.Prosessien

ja tuotteen laadunvarmistamisen avulla tarjotaan näkyvyyttä työprosessien suorittamiseen sekä niiden avulla tuotettuihin tuotteisiin.

Toinen CMMI:n mukainen lähestymistapa auditointien lisäksi on laadunvarmistamisen rakentaminen osaksi suoritettavaa työprosessia (laadun sisäänrakentaminen). Laadunvarmistustoimet perustuivat tällöin vertaisarviointiin. Erityistä huolenpitoa vaaditaan, että vertaisarvioinnin toteutuminen tapahtuu tarkoituksenmukaisesti. Jokaisen laadunvarmistukseen osallistuvan työntekijän tulee saada aiheeseen koulutus ja tuotteen arviointiin osallistuvat ovat projektin ulkopuolisia henkilöitä. Lisäksi laadunvarmistuksen tukemiseksi kootaan tarkastuslistoja joiden pohjalta havaitut puutteet tallennetaan.

Laadunvarmistaminen aloitetaan aikaisessa vaiheessa projektia, jotta saadaan aikaiseksi suunnitelmat, prosessit, menetelmät ja toimenpiteet. Näiden tarkoituksena on lisätä arvoa juuri kyseiselle projektille. Suunnitelmien tekemiseen osallistuvat ne henkilöt, jotka suorittavat laadunvarmistuksen toimenpiteitä. Toimenpiteet voivat perustua systemaattiseen tarkasteluun tai satunnaisotantaan, kuitenkin ne määritellään organisaation laatupolitiikan mukaisesti projektin luonne huomioiden. Havaitut tuotteen laatupoikkeamat käsitellään siten, jotta vastaavilta poikkeamilta vältyttäisiin tulevaisuudessa esimerkiksi puuttumalla työprosessin kulkuun [43].

4.7 Automaation laatutekijät

Automaatiosovelluksen keskeisimmät laatutekijät ovat turvallisuus, suorituskyky ja toimintavarmuus. Ohjelmistotuotannossa paineet tuotannon kehittämiseen ovat olleet suuremmat kuin automaatiosovellusten tuotannossa, koska automaation ja erityisesti sen ohjelmiston osuus teollisuuden kokonaistoimituksen kuluista on usein pieni, kuten johdannossa todettiin. Automaation laatu voidaan jakaa osatekijöihinsä. Kokonaislaadun muodostumiseen toiminnan oikeellisuuden lisäksi vaikuttavat monet muut tekijät. Toiminnan oikeellisuuden lisäksi tietoturva on merkittävä ja huomiota vaativa tekijä automaatiojärjestelmässä. Automaatiotoimittajan näkökulmasta palveluiden laadun osalta toimitusajat ovat tärkeä tekijä, jonka myötä asiakkaiden luottamus voidaan saavuttaa. Sovellussuunnittelijan näkökulmasta uudelleenkäytettävyys nousee merkittävämmäksi, koska tätä toivotaan löytyvän aikaisemmin toteutetuista projekteista koodin uudelleenkäytön takia.

Ohjelmistotuotannon menetelmät ovat pääsääntöisesti sovellettavissa automaation sovellussuunnitteluun, joten laadun tuottamiseen voidaan hyödyntää samoja ratkaisumalleja ja tekijöitä kuin ohjelmistotuotannossa. Näihin lukeutuvat laatua tukeva sovellussuunnitteluprosessi, oikeat teknologiset ratkaisut sekä riittävät resurssit ja motivaatio. Näitä vaaditaan korkeatasoisen ja luotettavan automaatiosovelluksen tuottamiseen [2, s. 113]. Sovelluksen laatu muodostuu tuotantoprosessissa täsmällisten ja huolellisesti suoritettujen osatehtävien avulla. Kypsymättömässä

suunnitteluprosessissa toiminta on suunnittelematonta, tilanteenmukaista tekemistä. Tällöin sovelluksen kehittäminen on muutamien kokeneiden suunnittelijoiden varassa, jolloin työntekijöiden vaihtuminen vaarantaa ohjelmiston ylläpidon sekä informaation välittymisen. Suunnitteluprosessin kypsyys ei mainita yksinään varmistavan tarvittavaa laatua, mutta kypsän prosessin eduksi nousee henkilöriippuvuuksien minimointi. Yksittäisten projektien henkilöitymistä tulisi pyrkiä välttämään. Lopputuotteen laadun hallitsemiseksi vaaditaan siis täsmällistä ja dokumentoitua suunnitteluprosessia [2].

Automaatiosovellusten kontekstissa laadunvarmistuksen tärkein todentamismenetelmä on dynaaminen testaaminen. Siinä toteutettu koodi testataan suorittamalla se varsinaisessa tai simuloidussa ajoympäristössä. Tämän rinnalla voidaan hyödyntää tarkastuksia ja staattista testausta. Ohjelmiston ja dokumentaation katselmointi voi koostua työryhmän tai vanhemman suunnittelijan ja toteutuksen suorittaneen nuoremman suunnittelijan vuorovaikutuksesta. Tällöin kyseessä ei tarvitse olla raskaita laatukomitean istuntoja. Mestari - kisälli periaatteella pystytään tuottamaan hyvälaatuisia automaatiosovelluksia hyvin kevyillä epämuodollisilla laadunvarmistustoiminnoilla, mutta mikäli asiakasta varten tarvitsee muodostaa todistusaineistoa laadun varmistamisesta, käytetään muodollisia menetelmiä. Viralliset katselmoinnit jäävät usein suorittamatta aikataulupaineiden tai mielenkiinnon puuttumisen vuoksi. Tällöin epävirallisen katselmoinnin hyödyt ovat huomattavat verrattuna tilanteeseen jossa toteutettua tuotosta ei ole missään vaiheessa tarkastettu [2].

Laadun ja luotettavuuden kannalta automaatiototeutuksen teknologiavalinnan pitää soveltua kohteeseen. Teknologialla tehtyjä toteutuksia on hyvä olla jo olemassa, tai että uusia ratkaisuja on koekäytetty riittävästi. Lisäksi valitun teknologian yleinen avoimuus ja standardien noudattaminen varmistaa sen, että ylläpidettävyys ja yhteensopivuusongelmia ei pääse tapahtumaan. Käytettyjen tuotteiden tuki tulee olla suhteutettuna järjestelmän suunniteltuun elinikään. Lopputuotteen laadun kannalta merkittävä tekijä on henkilöstön asiantuntevuus. Tähän sisältyy teknologian tuntemisen lisäksi aikaisempi kokemus sovelluskohteesta. Koska teknologiaa kehitetään jatkuvasti, on suunnittelijan osaamista ylläpidettävä. Uusimpien suunnittelutyökalujen käyttäminen ei itsessään paranna tuottavuutta tai laatua. Uudessa, juuri esitellyssä teknologiassa, on usein alkuvaiheessa tyyppivikoja jotka aiheuttavat ongelmia. Myöskään uusia ominaisuuksia ei välttämättä tunneta tarpeeksi hyvin, jotta niitä voitaisiin täysin hyödyntää [2].

4.8 Ohjelmistojen laatuterminologiaa

Ohjelmistojen laadun luonnehditaan muodostuvan dokumentoitujen toiminnallisuuksien ja suorituskykyä kuvaavien vaatimusten täyttämisestä. Näiden ulkoisten tekijöiden lisäksi laatu muodostuu kehitettävän sovelluksen implisiittisistä ominaisuuksis-

ta. Näihin ominaisuuksiin katsotaan kuuluvan yleiset laatuominaisuudet, joita odotetaan kaikelta ammattimaiselta sovelluskehitystyöltä. Dokumentoidut ohjelmistovaatimukset toimivat referenssinä joita vastaan toteutetun sovelluksen laatua verrataan laadunvarmistuksen toimilla. Ohjelmistojen ja sovellusten kehittämisessä eräs tärkeimmistä toimenpiteistä on testaaminen. Tätäkin merkittävämmän kokonaisuuden muodostaa varmistuminen siitä, että sovelluksen arkkitehtuuri ja toiminnallisuus on määritelty ja suunniteltu oikein [21].

Ohjelmistojen laadun arvioimiseksi muodostetussa standardissa ISO 9126 tarkastellaan tuotteen laatua eri näkökulmista. Standardissa esitetään laatumalli, joka jaottelee ohjelmiston laadukkuutta mittaavia kriteerejä. Näiden mukaan ohjelmiston laatuominaisuudet muodostuvat ohjelmiston toiminnallisuuksista, luotettavuudesta, käytettävyydestä, tehokkuudesta, ylläpidettävyydestä sekä siirrettävyydestä [41]. Edellä luetelluista laatuominaisuuksista luotettavuus voidaan jakaa vielä koostuvaksi sovelluksen toimintavarmuudesta, käyttövarmuudesta, ylläpidettävyydestä sekä toiminnan turvallisuudesta, että tietoturvasta [21]. Taulukkoon 4.1 on koottu laatuominaisuuksia joita sovelluksella voi olla. Termit esitetään, jotta eri lähteissä käytetyt samaa tarkoittavat termit on voitu tässä työssä yhtenäistää. Kaikki esitettävät laatuominaisuuksia koskevat asiat vastaavat taulukossa esitettyjä ja siitä voi tarkistaa niiden tarkoitetun merkityksen.

4.8.1 Laatupoikkeamat

Ohjelmistojen laaduntarkasteluun käytetään sisäisiä virhemekanismeja. Vaatimusmäärittelyssä tehdyt virheet näkyvät sovellussuunnittelussa ohjelmistovirheinä (error). Vaatimusten oikeellisuutta tai ristiriidattomuutta voi olla vaikea huomata. Ohjelmistovirheitä voi muodostua myös sovelluksen suunnitteluvaiheen aikana sekä sovellusta ohjelmoitaessa. Toteutus- ja koodausvirheiden löytämistä helpottaa testaamisen lisäksi järjestelmällinen ohjelmointi. Myös ohjelmistotyökaluissa ja käytettävissä valmiissa ohjelmistokomponenteissa voi esiintyä virheitä.

Ohjelmistovirheet ovat luonteeltaan systemaattisia. Virheen jäädessä huomaamatta, se voi epäsuotuisassa tilanteessa aiheuttaa sisäisen virhetilanteen (fault). Tilanteen kehittymisen laukaisee ihmislähtöinen väärä syöte tai ympäristöstä johtuva tapahtuma kuten laiterikko. Laiterikkojen yhteydessä laitteen vikaantumiseen voidaan vaikuttaa niin, että riskin pienentämiseksi laite vikaantuu turvalliseen suuntaan. Virhetilanne ei aina aiheuta virheellistä toimintaa. Mikäli näin kuitenkin tapahtuu, aiheutuu ulkoisesti havaittavissa oleva määrittelyn vastainen virhetoiminta (failure) [1, s. 119].

Kuva 4.1 *Ohjelmistotuotteen laatuominaisuuksia [32, s. 509-510]. Selitteiden suomennokset ovat vastaavat kuin opinnäytetyössä Ohjelmistotuotteen laadulliset haasteet [49].*

engl. termi	**suomenkielinen termi**	**selite**
correcteness	oikeellisuus	kattavuus, jolla ohjelma täyttää vaatimukset ja asiakkaan siihen kohdentamat tavoitteet
efficiency	tehokkuus	laskentaresurssien ja koodin määrä, jonka ohjelma tarvitsee tehtävänsä suorittamiseen
flexibility	joustavuus	työmäärä, joka tarvitaan ohjelman muokkaamiseen integrity yhtenäisyys kattavuus, jolla asiattomien pääsyä ohjelmaan tai sen sisältämään tietoon voidaan rajoittaa
interoperability	yhteentoimivuus	työmäärä, joka vaaditaan järjestelmän liittämiseksi toiseen järjestelmään
maintainability	ylläpidettävyys	työmäärä, joka vaaditaan ohjelmassa esiintyvän virheen paikantamiseen ja korjaamiseen (rajoitetusti määriteltynä)
portability	siirrettävyys	työmäärä, joka vaaditaan ohjelman siirtämiseen yhdestä laitteesta tai järjestelmäympäristöstä toiseen
reliability	luotettavuus	kattavuus, jolla ohjelman voidaan olettaa täyttävän tarkoituksensa vaaditulla tarkkuudella (epätäydellisesti määriteltynä, laajempia selitteitä on myös olemassa)
reusability	uudelleenkäytettävyys	kattavuus, jolla ohjelma (tai osa siitä) voidaan kierrättää muihin sovelluksiin, liittyen paketointiin ja ohjelmiston suorittamien toimintojen laajuuteen
testability	testattavuus	työmäärä, joka vaaditaan varmistamaan että ohjelma täyttää sille asetetut vaatimukset
usability	käytettävyys	työmäärä, joka vaaditaan ohjelman oppimiseen, operoimiseen, syötteiden valmisteluun ja tulosteiden käsittelemiseen

4.8.2 Laatumetriikka

Sovelluksesta mitattu laatumetriikan luotettavuusmetriikka ei anna yksityiskohtaista tietoa sovelluksen luotettavuudesta, mutta sitä voidaan hyödyntää sovelluskehitysprosessien parantamisessa tai tuotantolaitoksen luotettavuutta arvioitaessa. Laatumetriikkaa hyödynnetään erityisesti riskienhallinnassa. Laatumetriikkaa hyödynnetään kun tavoitteena on minimoida järjestelmän alhaalla olo aika sekä virhetoimintojen ja häiriöiden vaikutus. Keskimääräistä vikaantumisvälin käsite on yleisesti käytetty laitteiden ja niiden komponenttien, järjestelmien sekä tuotantolinjojen arvioimiseen.

Keskimääräinen vikaantumisväli (Mean Time Between Failure) muodostuu keskimääräisen vikaantumisajan (Mean Time To Failure) ja keskimääräisen toipumisajan (Mean Time To Restoration) summasta. Luotettavuudessa on kyse kohteen kyvystä suoriutua sille spesifioiduista sekä vaadituista tehtävistä määritellyissä olosuhteissa ja hyväksyttävänä aikana. Tuotantolaitoksen tapauksessa luotettavuutta arvioitaessa kaikki tuotantoon vaikuttavat komponentit otetaan huomioon, niin laitteiston kuin ohjelmiston osalta [21].

5. PROJEKTIN TUKITOIMINNOT

Projektin elinkaaren ajan kestävien tukitoimintojen avulla pyritään parantamaan projektinhallintaa sekä tuotteen laatua. Sovellussuunnitteluprojektien tapauksessa näihin tukitoimintoihin lukeutuu vaatimusten käsittely, laadunvarmistus sekä konfiguraation hallinta [12]. Edellä mainittujen tukitoimintojen lisäksi riskienhallinta on mukana ohjelmistoprojekteissa, erityisesti sellaisissa projekteissa joihin liittyy TLJ. Riskienhallinnan tehtävänä on riskien tunnistaminen, analysointi ja seuranta. Riskienhallintaa ei kuitenkaan käsitellä aihealueen rajauksista johtuen. Ensimmäisenä käsitellään edellisessä luvussa pohjustettu laadunvarmistaminen siihen liittyvine toimintoineen.

5.1 Laadunvarmistus

Laadunvarmistuksen toimenpiteillä voidaan tarkastella yritystoimintaa eri näkökulmista. Niillä voidaan käsitellä koko yrityksen toimia, laadunhallintajärjestelmää, projektin tapahtumia tai vaikka sovelluskehityksen yksittäistä vaihetuotetta. Laadukkaan prosessin tarkoitus on tuottaa kustannustehokkaasti halutunlaisia tuotteita. Standardi SFS EN 61506 mainitsee tärkeimmiksi sovellussuunnittelun laadunvarmistuskeinoiksi todentamisen ja kelpoistamisen, muutostenhallinnan sekä konfiguraationhallinnan [38].

Laadunvarmistus on kaikkien niiden suunniteltujen ja systemaattisten toimenpiteiden summa, joilla taataan automaation laatu [1, s. 30]. Laadunvarmistuksen tukitoimintoja hyödynnetään sovellussuunnitteluprojektin vaatiman luonteen mukaan. Laatutavoitteet voivat olla asetetut ohjelmistoprojektin laatusuunnitelman kautta, jolloin laatusuunnitelma määrittää koko projektin tai suunniteltavan sovelluksen laatutavoitteiden varmistamisen. Laatutavoitteet voidaan asettaa myös osana vaatimusmäärittelyä. Määrittelyvaiheen laadunvarmistukselliset toimet voivat olla myös määritelty laadunhallintasuunnitelmassa [22].

Laadunvarmistuksessa kokonaisvastuu on usein loppuasiakkaalla. Tämä ei kuitenkaan voi varmistua täysin toimittajan tuotteiden laadusta kun kyseessä on ohjelmistosovellus. Tästä syystä toimittajan velvollisuutena on tuottaa tarvittava aineisto, joka tukee asiakasta järjestelmän kelpoistamiseksi. Asiakkaan ja toimittaja laadunvarmistustoimet eroavat toisistaan. Toimittaja suorittaa testauksen, jolla se osoittaa oman työnsä laatua. Asiakas puolestaan toteuttaa laadunvarmistusprosessinsa mu-

kaisen kelpoistamisen jossa tarkastellaan toimittajan oman laadunhallintajärjestelmänsä noudattamisesta. Kelpoistamisen vastuualue jakaantuu ohjelmistoteollisuuden ja automaatioalan kohdalla hieman eri näkökulmasta. Automaatiotoimittajan laadunvarmistustoimet eivät lukeudu automaatiosovelluksen osalta kelpoistamisen piiriin, vaan kelpoistaminen rakentuu asiakkaan suorittamista toimenpiteistä. Toisin kuin ohjelmistoteollisuudessa, jossa ohjelmistosovelluksen toimittajan organisaatio kohdentaa myös kelpoistustoimenpiteitä kehitettävälle sovellukselle [1],[11].

Laadunvarmistusprosessiin kuuluvien toimintatapojen tulee olla systemaattisia ja prosessin tulisi olla hallittu ja mahdollisimman kevyt, sekä byrokratia on minimoitava. Koska yrityksen eri projekteissa prosesseille on hyvin erilaisia vaatimuksia, tulee laadunhallintajärjestelmässä olla joustavuutta. Dokumentoimalla joukko vaihtoehtoisia prosesseja, projektin luonne huomioiden voidaan valita projektin sovelluskohteeseen sopivin prosessi. Laadunvarmistusprosessin muodollisten toimintatapojen määrään vaikuttaa hyvin paljon asiakkaan vaatimukset (viranomaismääräykset). Muita prosessin luonteeseen vaikuttavia tekijöitä ovat hankkeen koko ja vaativuus, henkilöstön vaihtuvuus, henkilöstön kokemus ja ammattitaito sekä tuotteen luotettavuus ja muut laadulliset vaatimukset. Kokemattomat sovellussuunnittelijat tarvitsevat usein tiukan ja ohjeistetun suunnitteluprosessin ohjaamaan tekemistä. Vastavuoroisesti kokeneille ja ammattitaitoisille toteuttajille riittää löyhempi prosessi niin sovellussuunnittelun kuin laadunvarmistuksen osalta [11].

5.1.1 Todentaminen ja kelpoistaminen

Todentaminen ja kelpoistaminen (*verification and validation*) on tuotteen laadun tarkastelua, jossa tuotetta verrataan sen spesifikaatioon. Todentamisessa voidaan näyttää, että tuotteen toiminnallisten vaatimusten mukainen toteutus täyttää asiakasvaatimukset kohta kohdalta (jäljitettävyys) ja kelpoistamisella pyritään näyttämään että tuote vastaa asiakkaan tarpeita [12]. Todentamisen ja kelpoistamisen toimet kuuluvat osaksi sovellussuunnittelun laadunvarmistusta. Niiden tarjoamia laadunvarmistustoimia kohdistetaan kehitettävään sovellukseen koko sovellussuunnittelun ajan. Laadunvarmistuksen toimilla varmistutaan siitä, että soveltuva suunnitteluprosessi on määritelty ja siinä hyödynnetään parhaita käytäntöjä. Sovellussuunnitteluprosessilta vaadittavia asioita kuvataan standardissa ISO 90003 [44].

Todentamisella pyritään varmistumaan, että suunnitteluprosessi perustuu hyvään suunnittelukäytäntöön. Todentamisen toimintoja ovat tarkastaminen, katselmukset ja järjestelmän testaus. Todentaminen ja kelpoistaminen vaativat määrittelyn ja suunnittelun aikana toteutettavia spesifikaatioita, joiden perusteella laadunvarmistuksen toimenpiteet toteutetaan. Vaatimusmäärittelyn sisältämiä virheitä ei voida löytää kehitettävästä sovelluksesta vertaamalla sitä spesifikaation, jos itse spesifikaatio sisältää virheitä tai ei ole selkeästi kuvattu (yksiselitteisyys) [38].

Todentaminen ja kelpoistaminen perustuu asiakasvaatimusten ja järjestelmän toteutuksen välisen suhteen todistettuun olemassaoloon. Näiden välinen suhde, joka esiintyy jäljitettävyytenä, mahdollistaa todentamis- ja kelpoistamistoimien suorittamisen. Jäljitettävyys vastaa järjestelmävaatimusten linkittymisestä sovelluksen ominaisuuksiin. Sovelluskehityksen määrittelyvaiheessa todentamisen ja kelpoistamisen tehtävänä on selvittää, että kaikki järjestelmävaatimukset ovat jäljitettävissä vähintään yhteen ohjelmistovaatimukseen. Tämän tarkoituksena on varmistua, ettei järjestelmän ominaisuuksia ole unohdettu ohjelmistovaatimuksista. Vastaavasti selvitetään myös, että kaikki ohjelmistovaatimukset on linkitetty vähintään yhteen järjestelmävaatimukseen ja ettei sovellukseen olla kehittämässä määrittelyjen ulkopuolisia asioita [10].

CMMI-mallin todentamisen ja kelpoistamisen toimet liittyvät kypsyystason 3 vaatimiin toimenpiteisiin. Todentamisen toimenpiteet kohdennetaan tuotteeseen ja näiden avulla saadut tulokset tallennetaan ja analysoidaan. Näihin toimenpiteisiin luetellaan kuuluvan tässä työssä esiteltävät testaaminen, vertaisarviointina suoritettava katselmointi sekä formaali tarkastaminen. Kelpoistamisesta esitetään, että sen avulla varmistutaan tuotteen vaatimusten täyttymisestä. Kelpoistaminen tulee suorittaa tuotteella sellaisessa ympäristössä kun sitä tullaan käyttämään ja tapahtumaan voi osallistua loppuasiakas. Todentamisen ja kelpoistamisen toimenpiteistä saatujen tulosten perusteella identifioidaan ja implementoidaan korjaavat toimenpiteet [43].

Seuraavaksi tarkastellaan tässä esitetyt todentaminen ja kelpoistaminen yksityiskohtaisemmin sovellussuunnittelun kannalta, pääpainotus on sovelluksen toimittajan vastuualueeseen sijoittuvassa todentamisessa ja liittyvissä toimenpiteissä.

Todentaminen

Todentaminen esitellään tässä turvallisuuskriittisen järjestelmän näkökulmasta. Sovelluksen todentaminen suoritetaan kehityksen elinkaaren kaikissa vaiheissa ja todentamispolitiikka määritellään todentamissuunnitelmassa. Jokaisesta sovelluksen kehitysvaiheesta seuraavaan siirryttäessä tarkistetaan todentamisen menetelmin, että vaiheen mukaisen kehityksen tulos täyttää vaiheeseen liittyvät vaatimukset. Sovellussuunnittelun todentamisprosessi muodostuu siis vaiheen vaatimusdokumenttien vertaamisesta vaiheen lopputuloksena saataviin dokumentteihin tai koodiin [44, s. 36]. Todentaminen sisältää tuotteen tai vaihetuotteen verifioinnin kaikkien vaatimusten perusteella. Kyseessä on inkrementaalinen prosessi, joka tapahtuu pitkin tuotteen elinkaarta. Alkaen vaatimusten verifioinnista läpi kehityksen, aina valmiin tuotteen verifiointiin asti. Todentamisen menetelminä laadunvarmistuksessa käytetään usein vertaisarviointiin liittyviä toimenpiteitä kuten läpikäynti ja tarkastus. Sovellussuunnittelussa merkittävänä todentamistoimenpiteenä on sovelluksen tai sen

osan testaaminen [43, s. 401].

Todentamisen tuotteena saadaan vaiheen lähtötietojen vastaavuus vaiheen lopuksi tuotettuihin tuloksiin. Turvallisuuskriittisissä sovelluksissa ohjelmistovirheitä on aiheutunut virheellisten vaatimusmäärittelyjen takia. Tästä johtuen vaatimusmäärittelyt olisi erityisen tärkeää aina todentaa. Tämän tarkoituksena on varmistaa, että vaatimusmäärittely-spesifikaatio esittää sellaisia vaatimuksia järjestelmälle, jotta ne täyttävät asiakkaan osoittamat tarpeet. Nämä voivat sisältää asioita esimerkiksi projektin aloituspalaverin sisällöstä, asetetuista toteutettavuusvaatimuksista, viranomaismääräyksistä, yrityksen toimintatavoista ja politiikasta sekä loppukäyttäjän hyväksymiskriteereistä. Tämän vaiheen tärkein asia on varmistuminen siitä, että kaikki tarpeelliset toiminnot ja ominaisuudet, joita järjestelmältä vaaditaan, on kirjattu ja kuvattu selkeästi. Lisäksi todentamisen yhteydessä otetaan huomioon sellaisia seikkoja, kuten suunnittelun metodologia, aikaisemmat kokemukset sovelluskohteesta sekä järjestelmän ylläpidettävyydestä [44, s. 37].

Turvallisuuskriittisessä toteutuksessa seuraavakais todennetaan ohjelmistovaatimukset. Tähän vaiheeseen lukeutuu myös laitteistovaatimusten sekä testausvaatimusten verifiointi. Myös ei-toiminnalliset vaatimukset dokumentoidaan. Ohjelmistovaatimusten todentamisen tarkoituksena on varmistaa onko mitään edellisen vaiheen vaatimuksia jäänyt pois ja onko siinä esitetty kaikki vaatimukset. Todentaminen pitää sisällään ohjelmistovaatimusten analysoimisen täydellisyyden, johdonmukaisuuden, oikeellisuuden ja tarkkuuden osalta. Lisäksi varmistetaan ylläpidetyn jäljitettävyyden oikeellisuus [44, s. 39].

Ohjelmistovaatimusten todentamisen jälkeen verifioidaan Kaikkien vaatimusten tulee olla testattavia. Vaatimusten ja testitapausten välisen jäljitettävyyden paikkansapitävyys todennetaan. Tärkeäksi seikaksi nousee testaamisen kohdistaminen oikeisiin asioihin. Sovellusta tulee testata sitä vastaan mitä sen määrittelyjen mukaan tulee tehdä ja mitä ominaisuuksia siinä tulee olla [44, s. 40].

Järjestelmän käyttöönotto sisältää tarkastukset joiden avulla voidaan todeta että järjestelmä on asennettu oikein, sekä sen toiminnallisuudet ja suorituskyky ovat riittävät. Olennainen osa todentamista tässä vaiheessa on järjestelmän koekäytön ja testauksen kattavuudesta varmistuminen. Tällöin varmistetaan siis, että kaikki järjestelmän toiminnallisuudet on testattu käyttöönoton yhteydessä. Lisäksi voidaan myös varmistaa, että tehdastestin aikana simuloidut toiminnallisuudet antavat samat tulokset kuin asennetun järjestelmän toiminnallisuuksien testaus ja koeajo. Todentaminen keskittyy toimintojen oikea-aikaiseen ajoitukseen, suorituskykyyn sekä liityntöihin ulkoisiin järjestelmiin [44, s. 49-50].

Kelpoistaminen

Kelpoistamisen toimenpiteet kohdennetaan sellaisiin tekijöihin joiden avulla voi-

daan varhaisessa vaiheessa ennustaa kuinka hyvin tuote täyttää asiakkaan tarpeet. Kelpoistamisen avulla esitetään ollaanko valmistamassa oikeaa tuotetta, kun todentamisen menetelmillä varmistettiin valmistetaanko tuote oikein (v-mallin julkaisu, luku 3.2). Kelpoistaminen hyödyntää todentamisesta tuttuja menetelmiä. Kelpoistaminen suoritetaan usein loppukäyttäjän tai muun vastaavan sidosryhmän toimesta. Merkittävä asia on kelpoistusympäristö, jonka tulee vastata varsinaista tuotteelle ominaista ympäristöä. Aina kun mahdollista, kelpoistaminen suoritetaan lopullisessa tai sitä vastaavassa ympäristössä.

Kelpoistamisen ensimmäisenä vaiheena on valita kelpoistettava tuote tai sen osaalue. Kelpoistusmenetelminä voidaan käyttää asiakkaan toimesta tai kanssa suoritettavaa muodollista katselmointia, prototyypin demonstrointia sekä toiminnallisuuksien esittelyä. Kelpoistuksen valmistelussa huomioidaan kriteerien valitseminen siten, että kriteerien täytyttyä tuote vastaa tarkoitustaan. Seuraavaksi suoritetaan kelpoistaminen valituille tuotteille tai sen osille. Todentamisen aikana suoritettuja testitapauksia voidaan uudelleen käyttää kelpoistamisessa. Tällöin suunnitteluvaiheessa hyvin suoritettua testaussuunnitelmaa voidaan uudelleen käyttää kelpoistussuunnitelmana. Lopuksi analysoidaan kelpoistamisesta koostettuja raportteja havaituista ongelmista sekä tuotteeseen liittyvistä muutosehdotuksista. Analyysin tuloksena saatava informaatio kertoo kuinka tarvittavat muutokset voidaan toteuttaa, sekä kuinka tehdyt muutokset kelpoistetaan [43].

Kelpoistamista käytetään osoittamaan laatua. Osoittaminen tapahtuu tuotteen suunnittelemisen ja toteuttamisen rinnalla ja jatkuu järjestelmän koko elinkaaren ajan. Kelpoistamisella tarkoitetaan erityisesti asiakkaan vastuulla olevia laadunvarmistustehtäviä. Asiakas esittelee kelpoistustoimensa kelpoistussuunnitelmassa. Toimien aikana tarkastetaan, että hyvien käytänteiden mukaiset suunnittelu-, toteutussekä testaustavat ja tarkastukset on suunniteltu ja toteutettu dokumentoidusti [1, s. 26]. Projektissa jonka laatutavoitteet ovat korkealla, vaaditaan normaalia laajempi dokumentaatio. Automaatiojärjestelmän suunnittelun ja kehityksen aikana teknisten dokumenttien rinnan laaditaan kelpoistus- ja laatusuunnitelmien mukaisesti katselmointi ja testaussuunnitelmat. Kelpoistusvaiheessa asiakas arvioi kertyneen aineiston ja määrittelee sen pohjalta tarvittavat kelpoistustestaukset. Kokoamansa aineiston avulla kootaan järjestelmän kelpoistusraportti [1].

5.1.2 Läpikäynti

Läpikäynti (*walk-through*) on menetelmä jonka avulla tarkastetaan valmis tuote tai sen osa. Tarkastettaessa sovelluksen koodia sovellussuunnittelija kertoo miten luulee sovelluksensa toimivan. Tarkastaminen tapahtuu usein vertaisarviointina perehtyneemmän suunnittelijan toimesta ja kyseessä on usein epäformaali tilanne. Tarkoituksena ei ole käydä läpi koodattua toiminnallisuutta ja sen oikeellisuutta rivi

riviltä. Läpikäynti toimii arvioinnin lisäksi koulutus ja tiedonvaihto tilanteena. Läpikäyntitilaisuus voi siis muodostua kokeneemman suunnittelijan suorittamasta työn tarkastuksesta [12] tai läpikäynti voi olla formaali palaveri josta kirjataan pöytäkirja tai muistio [17].

Läpikäynnin aikana käsitellään kehitettävää sovellusta seuraavista näkökulmista [17]:

- etsitään poikkeavuuksia ja epäsäännönmukaisuuksia
- parannetaan tuotetta
- pohditaan vaihtoehtoisia toteutuskeinoja
- arvioidaan yhdenmukaisuutta

Sovelluksen suunnittelu- ja toteutusvaiheen aikana läpikäynnin avulla sovellussuunnittelijat vaihtavat ideoita ja saavat palautetta kollegoilta. Tämän seurauksena kehitettävää tuotetta pystytään parantamaan kaikin puolin. Koulutuksellisesta näkökulmasta se toimii metodina jolla saadaan kehitettyä nuorempien sovellussuunnittelijoiden tietotaito, tai kokeneempien suunnittelijoiden keskinäinen tietämys voidaan yhtenäistää ja nostaa samalle tasolle. Läpikäyntiin osallistuvilla tulisi olla hyvä perehtyneisyys joko sovelluskohteen alaan sekä/tai käytettyyn kehitystyökaluun ja teknologiaan.

Standardin IEEE 1028 [17] muodolliseen läpikäyntiin osallistuvien rooleiksi kuuluvat läpikäynnin vetäjä, sihteeri, suunnittelutyön tekijä sekä muut palaveriin kutsutut suunnittelijat, joiden tehtävänä on aineiston kommentointi. Vetäjän tehtäviin kuuluu palaverin sopiminen ja lähtöaineiston jakaminen ennen tapaamista sekä toiminnan järjestelmällinen ohjaaminen asetettujen tavoitteiden perusteella. Läpikäynnin aluksi jaetaan osallistujien roolit vetäjän toimesta, jonka jälkeen läpikäynnin kohde esitellään. Vetäjän tehtävänä on toimia puheenjohtajana ja huolehtia osallistujien aktiivisuudesta. Sihteerin tehtävänä on kerätä tallenteet läpikäynnin aikaisista huomioista ja kommenteista. Suunnittelun tekijä itse esittelee sovellustuotteen läpikäynnin aikana, johon muut palaveriin osallistuvat suunnittelijat esittävät kommentteja sekä kehitysehdotuksia. Näiltä osin formaali läpikäynti eroaa myöhemmin käsiteltävästä muodollisesta tarkastuksesta jossa palaverin aikana ainoastaan etsitään virheitä, eikä yhteisen ajan käyttäminen ratkaisujen miettimiseen ole sallittua.

Läpikäynnin suorittamisen tarpeellisuus arvioidaan projektin luonteen mukaan ja sitä varten varataan aikaa ja resursseja projektisuunnitelmaan. Ylimääräisiä läpikäyntejä suoritetaan yrityksen toimintatapojen mukaisesti, yleensä läpikäynti määrätään projekti- tai laatupäällikön toimesta, tai jos suunnittelija kokeen tämän olevan tarpeen. Läpikäyntitilanteessa lähtöaineistoksi tarvitaan läpikäynnille asetetut

tavoitteet, ohjelmistotuote jota tarkastellaan, standardit jotka vaikuttavat sovelluskohteen toimialaan ja sitä kautta sovelluksen kehittämiseen, standardit joiden perusteella sovellusta arvioidaan, havaittujen poikkeusten ja virheiden kategoriat joihin ne luokitellaan sekä tarkistuslista läpikäyntipalaverin aikana suoritettavista tehtävistä. Muodollinen läpikäynti päättyy kun asetetut tavoitteet on saavutettu, kommentit ja vaadittavat korjaukset on kirjattu ylös, mukaan lukien projektiin ja osallistujiin liittyvät tiedot, sekä kaikki vaaditut asiat on saatu kirjattua pöytäkirjaan. Pöytäkirja edustaa laadunvarmistuksen todistetta läpikäyntitalisuudesta [17, s. 29].

5.1.3 Tekninen katselmointi

Katselmointi (*review*) on yleinen termi ihmisten toimesta suoritetulle arvioinnille. Sovellussuunnittelussa katselmointi voidaan tehdä heti arvioitavan dokumentin tai koodin valmistuttua. Tässä on tarkoituksena etsiä vikoja ja epäjohdonmukaisuuksia sekä puutteita. Katselmoinnin hyödyksi usein mainitaan ongelmien löytyminen varhaisessa vaiheessa, jotta niiden vaikutukset eivät kertaudu projektin edetessä.

Tekninen katselmointi on formaali tilaisuus, jossa laadunvarmistuksellisina asioina katsotaan, että kaikki dokumentit ovat kunnossa ja kaikki vaaditut asiat suoritettu. Se on projektista ulospäin näkyvä tapahtuma, jolla tehdään projektin eteneminen näkyväksi. Katselmointi suoritetaan merkittävämpien tapahtumien päätteeksi, kuten sovellussuunnitteluvaiheen päätteeksi ennen käyttöönottovaiheeseen siirtymistä. Teknisen katselmoinnin voidaan luonnehtia olevan projektin hallinnallinen tapahtuma, eikä kyseessä ole niinkään laadunvarmistuksen toimenpide. Laadunvarmistuksessa on tarkoituksena löytää mahdollisimman paljon virheitä, kun taas teknisessä katselmuksessa löydetty puute tai virhe merkitsee huonosti tehtyä työtä [12].

IEEE standardin [17] mukaan teknisen katselmuksen aikana arvioidaan ohjelmistotuotteen soveltuvuutta tarkoitettuun käyttötarkoitukseen henkilöiden toimesta joiden ammattitaito on asiaan kuuluvalla tasolla. Tarkoituksena on tuottaa johdolle todisteita projektin teknisestä tilanteesta. Standardissa luetellaan sovellussuunnittelun elinkaaren aikaisia dokumentteja joita katselmukseen voidaan valita tarkastettaviksi. Lisäksi kuvataan tekniseen katselmukseen osallistuvien henkilöiden roolit sekä palaverin eteneminen yksityiskohtaisesti. Esitetyt käytänteet eivät eroa teknisen katselmuksen osalta formaalista läpikäynnistä, ainoastaan tässä asiakkaan mahdollinen läsnäolo tapaamisessa poikkeaa läpikäynnistä. Standardissa mainitaan myös palaverin aloitus kriteerit ja tarvittavat dokumentit, sekä lopetuskriteerit ja tuotettavat dokumentit. Näitä ei kuitenkaan luetella tässä, koska aikaisemmin todettiin olevan kyse projektinhallinnallisesta tapahtumasta, jotka on pyritty rajaamaan aihealueen ulkopuolelle.

5.1.4 Testaaminen

Sovellussuunnittelun laadunvarmistuksesta ei voida puhua ilman että mainitaan koodin testaamista. Tässä kappaleessa käydään pintapuolisesti läpi sovelluksen testaamiseen liittyviä asioita. Testaamisen tarkoituksena on taata, että kehitettävä sovellus vastaa spesifikaatiota. Käytettäessä v-mallin mukaista elinkaarimallia, testaaminen aloitetaan toteutusvaiheen aikana ja jatketaan aina lopulliseen hyväksyntään eli hyväksyntätestaukseen asti. Testauksen vaiheet voidaan jakaa moduuli-, integrointi- ja järjestelmätestaukseen. Testaustoimenpiteet muodostuvat staattisesta ja dynaamisesta testaamisesta. Dynaamisessa testaamisessa sovellusta testataan suorituksen aikana. Staattisessa testaamisessa sovellusta ei suoriteta ollenkaan, vaan ainoastaan koodia tarkastellaan automaattisilla tarkastustyökaluilla tai ihmisten toimesta.

Reaaliaikajärjestelmien testauksessa ongelmia tuottaa sovelluskohteen reaalimaailman tilanteiden simulointi, jossa eri ajokerroilla tapahtuvat ulkoisten vaikutusten ajoitukset vaihtuvat. Virhetilanteiden löytäminen simuloimalla muodostuu mahdottomaksi ja ohjelman testauksessa pystytään tutkimaan vain pieni lukumäärä kaikista mahdollisista ulkoisten vaikutteiden tilanteista. Tämän takia pelkkään simulointitestaukseen ei kannata käyttää ylimääräistä aikaa, vaan jakaa testaukseen käytettävä aika kaikkien testausmuotojen kesken.

Ainoastaan sovelluskohtaisesti toteutettuihin ohjelmamoduuleihin kohdistetaan todentamisen ja kelpoistamisen mukaisia testaustoimia. Ohjelmistoarkkitehtuurin laitetoimittajan tarjoamiin standardilohkoihin ei ole syytä kohdentaa tässä esiteltyjä todentamisen toimenpiteitä [44]. Lisäksi yrityksen oman tuotekehityksen tuloksena kehitettyjä ohjelmalohkoja ei erikseen testata, koska nämä ovat useissa projekteissa käytössä ja jo hyvin testattuja. Kuitenkin piirikokonaisuudet joissa kaikkia edellä mainittuja lohkoja käytetään, testataan järjestelmätestauksessa sekä uudestaan koeajojen yhteydessä.

Moduulitestaus tarkoittaa yksittäisen osakokonaisuuden testaamista. Ohjelmistotuotannossa kyseessä on luokka tai komponentti, automaatio ja ohjausjärjestelmien tapauksessa kyseessä on toiminnallisuuden toteuttava sekvenssi tai toimilohko. Testaamisen tarkoituksena on todentaa, että kyseinen koodi toimii oikein verrattuna testitapauksessa esitettyyn odotettuun toimintaan. Ohjelmistotuotannossa testitapaukset suunnitellaan suunnitteluvaiheessa tehtyjen käyttötapausten perusteella. Niiden tarkoituksena on asiakasvaatimusten toteutumisen varmistaminen. Ohjelmistotuotannossa yksikkötestaukseen käytetään tavallisesti lasilaatikkotestaamista (glass box-, white-box testing), jolloin testaajalla on pääsy ohjelmakoodiin. Tällöin testauksen kohteena on testattavan moduulin sisäinen rakenne, ei toiminta [47].

Automaatio ja ohjausjärjestelmien moduulitestaaminen muodostuu pääosin ohjauksia toteuttavien, sekä laskusuorituksia toteuttavien toimilohkojen testaamises-

ta. Ohjauksia toteuttavien ohjelmalohkojen testaamisessa varmistetaan toimintakuvauksen mukainen toiminnallisuus. Tällöin toimilohkon syötteinä on toimilaitteiden, mittausten ja muiden ohjelmalohkojen tilat, sekä toiminnallisuuksien ohjaamiseen hyödynnettävät aseteltavat parametrit. Syötettyjen asetusarvojen laillisuus rajoitetaan usein toimilohkojen ulkopuolisen valvomosuunnittelun keinoin, eikä yksikkötestauksen raja-arvoanalyysiä yleensä suoriteta lohkomoduulia testattaessa. Moduulitestaus kohdennetaan ainoastaan kehitettyihin lohkoihin tai muutettuihin. Moduulitestausympäristönä käytetään kehitystyökalun simulointiympäristöä tai suorittamalla testattavat lohkot varsinaisessa ohjelmoitavassa logiikassa.

Integrointitestaus sisältää valmiiden ja testatuttujen moduulien yhteenliittämisen testauksen. Siinä testataan toisiinsa liitettyjen moduulikohtaisten rajapintojen yhteistoimintaa. Integrointi toteutetaan jatkuvalla integroinnilla, mikrosykleittäin tai kaikki moduulit yhdellä kertaa (big bang -integrointi). Jatkuvassa integroinnissa kokonaisuus kootaan yksikkö kerrallaan aiemmin testatuista moduuleista. Kokoavassa bottom-up integroinnissa aloitetaan yksiköiden riippuvuushierarkian pohjalta ja edetään ylöspäin. Jäsentävässä top-down testauksessa aloitetaan hierarkian päältä ja edetään alaspäin. Tällöin vielä toteuttamattomat yksiköt pitää olla korvattuna niitä edustavilla tynkätoteutuksilla [21].

Integrointitestausta ei välttämättä tehdä erikseen, jos uuden moduulin valmistuttua se liitetään se kokonaisuuteen joka testataan tämän jälkeen. Integrointitestauksessa usein hyödynnetään mustalaatikkotestausta (*black box*), jossa testaajalla ei ole tarkkaa tietoa toteutuksen sisäisestä rakenteesta. Sen tarkoituksena on tarkastella että kehityksen kohteena olevalla sovelluksella on oletetun mukainen toiminta ja suorituskyky [47]. Integrointitestaus käsittää kehitettävän sovelluksen moduulien ja komponenttien integroimisen valmiiksi sovellukseksi, mutta kirjallisuuslähteestä riippuen se voi käsittää myös laitteiston ja ohjelmiston integroimisen.

Järjestelmätestaus käsittää ohjelmiston ja laitteiston yhteistestauksen. Kokonaista sovellusta testattaessa tarkastellaan myös ei-toiminnallisista ominaisuuksista esimerkiksi suorituskykyä. Tästä johtuen testaaminen täytyy suorittaa varsinaisella laitteistolla, jossa ohjelmistoa tullaan ajamaan [47]. Tämä tapahtuu ennen tehdastestausta ja toimittajan tiloihin asennetulla järjestelmällä ja kyseessä on siis koko tuotteen testaaminen [38]. Testaamisen kohteena on erityisesti kehitetyn automaatiosovelluksen testaaminen varsinaisessa laitteistossa. Tällöin järjestelmä muodostuu esimerkiksi ohjelmoitavasta logiikasta, tähän kytketyistä muista laitteista ja järjestelmistä, sekä varsinaisesta tai virtualisoidusta käyttöliittymästä. Tämän testauksen tarkoituksena on testata käyttöliittymärajapinta sekä I/O- ja väylärajapinta. Lisäksi toiminnallisuuksien testaaminen mahdollistuu käyttöliittymän kautta, jossa toimintojen ilmaisuvoima edesauttaa virheiden havaitsemista. Käyttöliittymän graafisesta esityksestä on havaittavissa toiminnallisuuksien virheet huomattavasti kehitysym-

päristön sekvenssi- ja tikapuukaavioiden sekä syötteiden ja tulosten tarkastelusta ajonaikana.

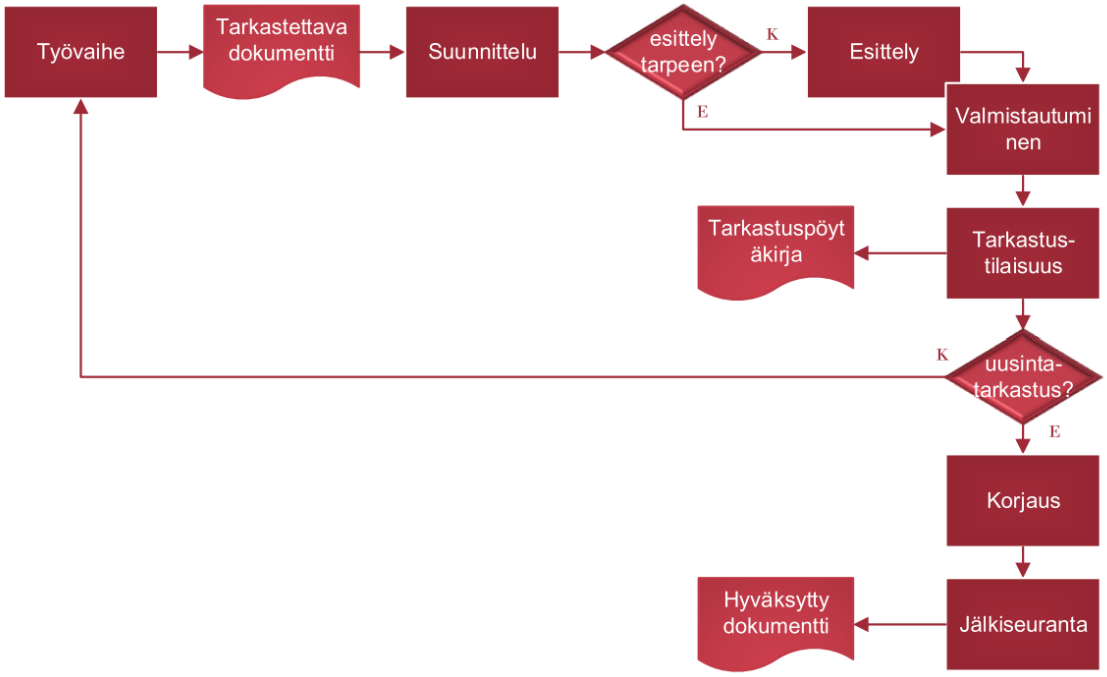
Toimilaitteisiin liittyvät testit on syytä suorittaa uudelleen automaatiosovelluksen käyttöönoton testausvaiheissa, koska järjestelmätestauksessa mukana ei ole toimilaitteiden ja kohdeprosessin dynamiikkaa. Järjestelmätestauksessa ainoastaan toimilaitteita simuloidaan ohjelmallisesti sekä ulkoisilla simulointisignaaleilla.

5.1.5 Tarkastus

Ohjelmistosovelluksen tarkastus (*inspection*) on esitelty parantamaan kehitettävän ohjelmistotuotteen laatua ja kohentamaan ohjelmointityön tuottavuutta [9]. Tarkastus on muodollinen katselmointi, joka tapahtuu määriteltyjen sääntöjen mukaisesti ja sitä luonnehditaan ohjelmistotuotannon laadunvarmistuksen keskeiseksi työvälineeksi ja tehokkaimmaksi menetelmäksi. Tarkoituksena on tarkastaa ohjelmistoprojektin dokumentti tai ohjelmakoodi vertaamalla sitä spesifikaatioon. Kyseessä on siis puhdas laadunvarmistuksen todentamisen toimenpide, jonka tarkoituksena on tarkastaa tiettyyn vaiheeseen suoritettu osakokonaisuus. Tarkastustilaisuudessa voidaan tarkastaa kaikki suunnitteluprojektissa tehty työ, mutta tarkastukset kohdistetaan usein määrittelydokumentteihin joita ei voida ohjelmakoodin tapaan erikseen testata [12].

Alkuperäisen vuoden 1976 julkaisun *Design and code inspections and process control in the development of programs* [9] mukaan, jossa tarkastaminen esitellään, ohjelmistotuotteeseen päässeet virheet olisi löydettävä mahdollisimman aikaisessa vaiheessa ja lähellä niiden syntymistä. Virheiden korjauskulut kasvavat sitä suuremmiksi, mitä myöhemmässä vaiheessa ohjelmistoprojektin elinkaarta ne havaitaan. Tähän vaikuttaa minkä elinkaarimallin mukaan sovellusta kehitetään, eli missä vaiheessa suoritetaan sovelluksen testaus.

Tarkastus tarjoaa systemaattisen lähestymistavan ohjelmistotuotteen yksityiskohtaiseen tarkasteluun. Toimenpide on tarkoitettu tuotteen laadun varmistamiseen, eikä sovellussuunnittelun prosessin tarkasteluun. Tarkastus asettuu laadunvarmistustoimena testaamisen ja formaalin katselmoinnin välimaastoon. Se tarjoaa suunnitteludokumentaation tarkastamisen lisäksi mahdollisuuden keskeneräisen ohjelmakoodin tarkasteluun, jota ei voida testata ajon aikaisilla dynaamisilla testausmenettelyillä. Tarkastamisen kohteiksi valitut kriteerit voivat liittyä esimerkiksi koodin laatuominaisuuksiin kuten toimintojen oikeellisuus, tai ohjelmointiteknisiin asioihin jotka vaikuttavat ylläpidettävyyteen ja käytettävyyteen kuten ohjelmointityyli, muuttujien ja tagien nimeäminen sekä kommentoinnin asiallisuus [29]. Jälkimmäiseksi mainittujen tekijöiden varmistaminen voidaan suorittaa ainoastaan tarkastamisen tai läpikäynnin avulla. Erityisesti aloittelevan sovellussuunnittelijan tuottama koodi on tarpeellista tarkastaa aikaisessa vaiheessa [9].



Kuva 5.1 *Tarkastusmenettelyn prosessikaavio [41].*

Tarkastukseen osallistuvien pitää tutustua tarkastettavaan materiaaliin kunnolla etukäteen, jotta tarkastustilaisuudesta saadaan sen tarjoama hyöty. Muuten tilaisuus jää tehottomaksi, eikä tarkasteltavasta materiaalista löydetä puutteita. Kaikissa tarkastuksissa tulee olla paikanpäällä niin vähän henkilöitä kuin todella tarvitaan. Yksinkertaisesti tilaisuuteen osallistujilla tulee olla jotain sinne annettavaa, tai jotain siitä saatavaa ja tilaisuuden tulee olla tehokas jotta se ei vie turhaan työaikaa. Tarkastustilaisuuden eteneminen kuvataan vaihe vaiheelta kuvassa 5.1.

Tarkastuksen vaiheet

Tarkastusten vaiheet esitellään kuvan 5.1 prosessikaaviossa. Siinä vaiheina ovat tarkastusten suunnittelu, valmistautuminen, tarkasteltavan kohteen esittely, tarkastustilaisuus, löydettyjen virheiden korjaaminen ja jälkiseuranta. Tavoitteena on työvaiheen aikaiselle tarkastettavalle dokumentille hyväksynnän saaminen tarkastusmenettelyn avulla. Tarkastusten suunnittelu otetaan mukaan osaksi projektisuunnittelua ja tarkastuksia varten tulee varata aikaa, mutta ajankohtia ei voida ennakoida. Tilaisuudet suoritetaan esiintulleiden tarpeiden ja työn edistymisen perusteella [9].

Tarkastusryhmän muodostaa puheenjohtaja, suunnittelutyön tekijä, sihteeri ja alustaja. Lisäksi mukaan tulisi ottaa erityisalojen asiantuntijoita tarpeen vaatiessa. Tarkastusmenettelyn toimintaa kuvataan sosiaaliseksi ja projektinhallinnalliseksi tapahtumaksi. Ryhmän kaikki jäsenet toimivat tarkastajina, mutta erilliset roolit voidaan sopia etukäteen. Tällöin tarkastellaan materiaalia käyttäjän, ylläpitäjän tai

testaajan näkökulmasta. Lisäksi tarkastajalla voi olla tarkastusten kohteena standardien ja menetelmien valvonta. Kuvaus tarkastusryhmän jäsenistä ja niiden tehtävistä löytyvät alkuperäisestä julkaisusta sekä IEEE 1028 standardista.

Esittelyssä eli alustustilaisuudessa suunnittelija esittelee tarkastettavan materiaalin. Esittelijä käy läpi kokonaisuuden johon tarkastettava materiaali kuuluu. Tämän jälkeen hän tarkentaa esittelyn koskemaan hänen suunnittelemiaan osuuksia suuremmasta kokonaisuudesta. Tekijä jakaa materiaalin tarpeeksi ajoissa ennen tarkastuksen suorittamista. Tekijän huolehdittavaksi myös jää pohjatöiden teko. Tällöin tarkastukset eivät työllistä tarpeettomasti muita tarkastusryhmään kuuluvia jäseniä. Erityisen tärkeää on, että tarkastettava materiaali on hyvin tehty. Muuten tarkastuksesta tulee helposti keskeneräisen työn ideapalaveri jossa esitetään ratkaisuja työn tekemiseen, joka ei ole tarkastustilaisuuden tarkoitus [12],[9].

Tarkastusmenettelyn prosessikaaviossa olevalla valmistautumisella tarkoitetaan osallistujien tutustumista tarkastusta varten etukäteen jaettuun materiaaliin. Ennen tarkastusta tarkastajat pyrkivät löytämään mahdollisimman paljon ongelmakohtia. Kaikkien tarkastukseen osallistuvien tutustuttua materiaaliin etukäteen on heillä kommentit kirjattuna jo valmiina. Tarkastukseen osallistuvat luokittelevat löytämänsä virheet etukäteen ilmoitettuihin virhetyyppikategorioihin. Tarkastamisen avuksi materiaalin mukana toimitetaan tarkastuslistat [1], jotka ohjaavat tarkastuksen oikeisiin asioihin [9].

Tarkastettava materiaali joko hyväksytään tai päätetään uudesta tarkastuksesta. Prosessikaavion loppuun sijoitettu korjaaminen sisältää tarkastustilaisuudessa ylös kirjattujen virheiden ja puutteiden korjaamistyön. Korjaustoimien virheettömyyttä pidetään usein itsestään selvänä, mutta näin asia ei kuitenkaan ole. Tämän takia viimeiseksi tarkastuksen toiminnoksi on mainittu jälkiseuranta jonka tarkoituksena päästä eroon huonoista korjauksista jotka itsessään eivät toimi, tai aiheuttavat uusia vikoja [9].

Tarkastusten käyttöönotto vaikuttaa kokonaisvaltaisesti työntekokulttuuriin. Suunnittelun tekijä suoriutuu työstään täysin toisella tavalla, kun hän tietää että työlle tulee julkinen tarkastus. Tarkastukset ovat arkaluontoinen prosessi, joka vaatii organisaatiokulttuurilta kypsää suhtautumista. Tarkastuskulttuurin kehittyessä asenteet muuttuvat vähitellen ja tarkastusten tarkoitukseksi aletaan kokea tekijän auttaminen ja tarpeellisen laadun aikaansaaminen lopputulokseen. Tarkastustilaisuuksien myötä suunnittelija saa käsityksen siitä, minkä tyyppisiin virheisiin hän on taipuvainen. Kun ohjelmointityö tarkastetaan aikaisessa vaiheessa projektia, tarjoutuu suunnittelijalle mahdollisuus oppia virheistään ja osoittaa kehittyvänsä myöhempien tarkastusten aikana [12],[9].

[1] Liitteenä A on automaatiosovelluksen tukidokumentaation dokumenttipohjaksi kehitetty tarkastuslista.

Tarkastukseen osallistuvilla projektin työntekijöillä on aiheen esittelyn, valmistautumisen ja tarkastustilaisuuden jälkeen yhtenäisempi käsitys projektin sovelluskohteesta. Tämän katsotaan lisäävän tarkastukseen osallistuneiden kaikkien henkilöiden tietotaitoa jonka luonnollisesti katsotaan kasvattavan tuottavuutta. Projektinhallinnan kannalta tarkastukset luovat ohjelmistokehitykselle välitavoitteita. Näin työ jakaantuu osakokonaisuuksiin ja sovelluskehityksen aikataulussa pysymisen seuranta kohenee. Tarkastusryhmä ottaa kantaakseen vastuuta sovelluskehityksen sisällöstä ja suunnittelija voi olla varmempi työnsä oikeellisuudesta [12],[9].

5.1.6 Tarkastuksen eroavaisuudet tekniseen katselmointiin

Teknisen katselmuksen ja tarkastuksen avulla voidaan tarkastella ohjelmistotuotetta tai dokumenttia sen soveltuvuudesta tarkoitettuun käyttöön. Seuraavassa vertaillaan edellä esitettyjä tilaisuuksia. Molemmissa jaetaan tarkasteltava materiaali etukäteen tapaamiseen osallistujille, joiden tulee valmistella näihin liittyvät kommentit valmiiksi. Tämän lisäksi tarkastustilaisuuden aikana on tarkoitus löytää yhdessä uusia epäkohtia materiaalista, etukäteen löydettyjen lisäksi. Tekniseen katselmukseen voidaan kutsua osallistumaan johtoa tai asiakkaan edustajia toisin kuin tarkastustilaisuuteen, johon osallistuvien henkilöiden määrä pyritään pitämään niin pienenä kuin mahdollista.

Teknisen katselmuksen kulku tulee noudattaa etukäteen ilmoitettua asialistaa sekä tilaisuudesta tulee tehdä tallenteita. Tarkastustilaisuudessa puolestaan tulee noudattaa jaettujen roolien mukaisia toimia, sekä tilaisuuden puheenjohtajalla tulisi olla formaalin tarkastuksen pitämiseen asianmukainen perehtyneisyys. Tarkastustilaisuuden aikana suoritettavaa tarkastamista varten suositellaan tarkastettavaan materiaaliin soveltuvaa tarkastuslistaa. Standardissa määritellään myös ohjelmistotuotannon tuotteen tarkastusprosessin ajankäyttö tarkastettavaa koodiriviä kohden sekä tarkastustilaisuuden kokonaiskesto.

Tarkastustilaisuuden alkaessa tarkastuksen puheenjohtajan havaitessa, että tarkastajat eivät ole valmistautuneet, siirretään tapaamista. Teknisen katselmuksen alkamiseen tai päättymiseen ei ole esitetty yksityiskohtaisia vaatimuksia IEEE:n standardissa. Katselmuksen päätyttyä tehdään muodollinen katselmusraportti. Myös tarkastustilaisuuden päätteeksi tehdään muodollinen raportti, mutta tämän lisäksi tulee koota yhteenveto havaituista epäkohdista. Jaottelemalla havaitut epäkohdat kategorioihin, voidaan nämä tilastoimilla saada mittareita työprosessien mittaamiseen ja kehittämisseurantaan. Tarkastettavan materiaalin osalta tarkastus katsotaan päättyneeksi vasta kun havaitut poikkeamat on korjattu ja materiaali on hyväksytty.

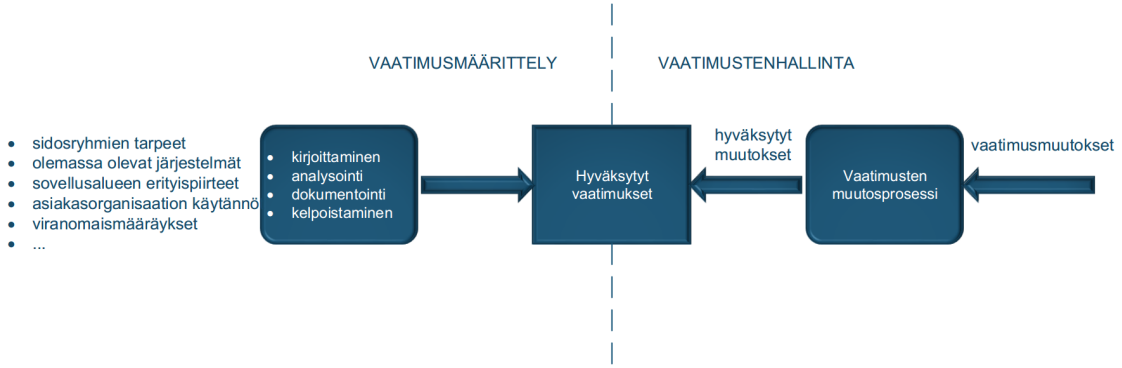
5.2 Vaatimustenkäsittely

Vaatimustenkäsittely on perusedellytys sovellusprojektin onnistumiseen. Vaatimukset ovat jotain mitä tuotteella pystytään tekemään, tai ominaisuus joka tuotteella tulee olla [39, s. 83]. Ohjelmistosovelluksen vaatimukset voidaan jaotella toiminnallisiin, ei-toiminnallisiin sekä sovelluksen toteutusta rajoittaviin reunaehtoihin. Suoraan asiakkaan tarpeista tulevat vaatimukset ovat asiakasvaatimuksia. Asiakasvaatimuksissa kuvataan projektin sisältö asiakkaan kannalta ja ne lähtevät asiakkaan liiketoiminnan tarpeista. Sommerville kuvailee kirjassaan *Software Engineering* [39] vaatimusten olevan luokiteltu ylemmäntason vaatimuksiin, joihin asiakasvaatimukset lukeutuvat sekä alemmantason järjestelmävaatimuksiin, joissa esitetään yksityiskohtaisesti mitä kehitettävän järjestelmän tulee tehdä. Abstraktit asiakasvaatimukset kuvataan kaavioilla sekä luonnollisella kielellä ja kuvaamiseen käytetään sovelluskohteen käsitteistöä. Yksityiskohtaiset järjestelmävaatimukset, voidaan käyttää tässä nimitystä toiminnallinen määrittely tai toimintakuvaus, ilmaisevat mitä järjestelmää toteutettaessa tulee implementoida [39].

Vaatimustenkäsittely koostuu vaatimusten määrittelystä ja vaatimusten hallinnasta. Näiden välinen työnjako on esitetty kuvassa 5.2, jossa näiden avulla aikaansaadaan hyväksyttyjä vaatimuksia. Vaatimusmäärittelyssä kartoitetaan vaatimukset sovelluskohteen sidosryhmien avulla. Tämä voi rakentua omaksi projektikseen jonka aikana selvitetään asiakkaan vaatimukset. Vaatimusmäärittelyyn sisältyvä vaatimusten kartoittaminen voidaan suorittaa esimerkiksi tulevia käyttäjiä haastattelemalla. Vaatimuksia tarkennetaan niiden tärkeyden ja keskinäisten riippuvuuksien osalta vaatimusanalyysissä. Vaatimustenkäsittely on sovelluksen laadun kannalta tärkeä, koska laatu määritellään sen mukaan kuinka hyvin tuote vastaa vaatimuksia. Tuotteen ja vaatimusten vertaamiseen käytetään laadunvarmistuksen menetelmiä. Mikäli vaatimusmäärittelyssä on puutteita, ei tällöin voida havaita laatupoikkeamia vertaamalla tuotetta sen määrittelevään spesifikaatioon. Tämän takia myös vaatimusmäärittely tulee asettaa laadunvarmistuksen piiriin todentamisen ja kelpoistamisen menetelmin [11].

Vaatimusten kelpoistamisessa varmistetaan, että kuvatut vaatimukset todella esittävät sellaista järjestelmää jonka asiakas haluaa. Kelpoistaminen asettuu osin päällekkäin vaatimusanalyysin kanssa, joiden molempien avulla etsitään virheitä. Vaatimuksia voidaan kelpoistaa katselmoimalla määrittelydokumentaatio sekä muodostamalla vaatimuksia vastaava testaussuunnitelma. Mikäli testitapauksien muodostaminen on ongelmallista, on tämä heijastumaa huonosti määritellyistä vaatimuksista. Oikein kuvattu vaatimus on aina testattavissa [39].

Vaatimustenhallinnalla pidetään kirjaa sovellusvaatimuksista sekä käsitellään projektin aikana vaatimuksiin tulevia muutoksia. Vaatimuksiin tulee usein muutoksia



Kuva 5.2 *Vaatimustenkäsittelyn muodostuminen vaatimusten määrittelystä ja hallinnasta [11].*

pitkin projektia, joka voi johtua asiakkaan yrityksen lähtökohtien muuttumisesta ajan myötä. Myös toteutuksen yksityiskohdat voivat aiheuttaa muutoksia, sillä kaikkien yksityiskohtien huomioonottaminen ennen toteutusta on hankalaa. [39, s. 43] Ohjelmistotuotteen vaatimustenkäsittely on lähellä muutostenhallintaa. Työnjako näiden välillä tulee määritellä. Esimerkiksi pienissä projekteissa vaatimustenhallinta hoidetaan kevyellä prosessilla ja vasta ylläpitovaiheessa siirrytään käyttämään formaalimpaa ja raskaampaa muutostenhallintaa [11].

CMMI kypsyysmallin tasolla 2 (toistettava) vaatimustenhallinnalla huolehditaan, että koko projektin ajan ylläpidetään vaatimusmäärittelyllä hyväksyttäviä ja ajan tasalla olevia vaatimuksia. Vaatimustenhallinnan tehtävänä on myös huolehtia siitä, että vaatimusten ja tuotteen suunnittelun välillä ei ole epäjohdonmukaisuuksia [43].

5.2.1 Vaatimusmäärittely

Automaatioprojektissa määrittelyvaiheen aikana muodostetaan kuvaus suunniteltavasta järjestelmästä. Tuotetta kehittävä organisaatio esittää mitä järjestelmän tulee tehdä, riippumatta miten toiminnot toteutetaan. On tärkeää että muodostettu spesifikaatio on paikkansapitävä asiakasvaatimusten kanssa. Tämän toteutumiseksi vaatimusmäärittelyt toteutetaan jäljitettävästi. Tämän avulla on mahdollista osoittaa kuinka vaatimusmäärittelyn yksityiskohtaiset kuvaukset liittyvät yleistasoisiin asiakasvaatimuksiin. Asiakasvaatimusten selvittämisen ohella vaatimusmäärittelyssä analysoidaan ja kelpoistetaan vaatimukset [28].

Vaatimusmäärittelyspesifikaatiota muodostettaessa voidaan käyttää erilaisia spesifiointimenetelmiä. Vaatimustenhallinta ja -määrittely liittyvät kiinteästi toisiinsa. Määrittelytoiminnolla tuotetaan dokumentteja vaatimustenhallinnan käyttöön, jonka tehtävänä on käsitellä ja ylläpitää niitä. Vaatimusmäärittelyn laadintaan kirjataan yleisen tason asiakasvaatimukset, keskeiset asianomistajat, sovelluksen rajaus,

toteutuksen rajoitukset ja halutut tuotepiirteet. Tämän jälkeen tarkastetaan että nämä ovat ymmärrettävissä [21]. Dokumentoidut vaatimukset katselmoidaan yhdessä asiakkaan kanssa. Vaatimusmäärittelyn tuloksena syntyy toiminnallinen määrittely, joka pitää sisällään tavallisesti asiakas- ja ohjelmistovaatimukset. Määrittelyn yhteydessä voidaan luonnostella asiakasvaatimuskohtainen testaussuunnitelma. Dokumentointiin voidaan hyödyntää wiki-alustaa, jossa kaikki kyseiseen vaiheeseen osallistuvat työntekijät muodostavat dokumentin yhteisvoimin [12].

Käyttötapaukset ovat UML-mallinnuskielinen menetelmä vaatimusmäärittelyjen kuvaamiseen. Niiden avulla mallinnetaan kehitettävän sovelluksen toiminnalliset vaatimukset ja ne antavat suunnittelijalle rakenteellisen käsityksen järjestelmän toiminnallisista vaatimuksista. Käyttötapaukset muodostuvat järjestelmän tai sovelluksen käyttöön liittyvistä kokonaisuuksista, joilla on merkitystä järjestelmän ulkopuolisten tekijöiden kannalta. Ulkoisiin toimijoihin katsotaan kuuluvan ihmiset, laitteet, sekä muut järjestelmät ja sovellukset joiden kanssa järjestelmä on vuorovaikutuksessa. Käyttötapauskaaviot ovat eräänlainen kartta kehitettävästä sovelluksesta vaatimusten näkökulmasta. Käyttötapauksiin jaetun suunnitelma katsotaan myös edesauttavan ohjelmoijaa ajankäytön suunnittelussa sekä osaltaan toteuttavat asiakasvaatimusten jäljitettävyyttä [21],[40].

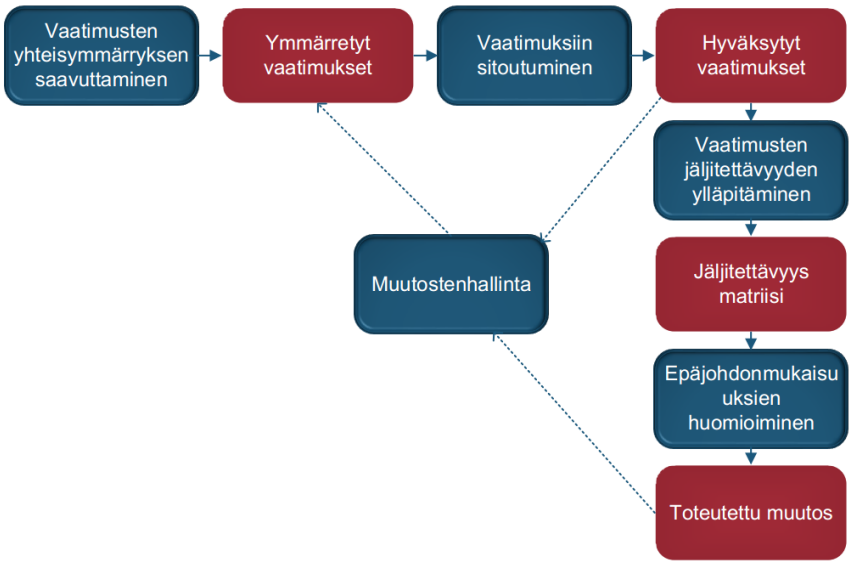
5.2.2 Vaatimustenhallinta

Vaatimustenhallinnalla pyritään varmistamaan, että lopputuote vastaa asiakkaan vaatimuksia. Asiakasvaatimusten muuttuminen kesken projektin todetaan olevan tavallista [12],[49]. Muutos voi johtua markkinatilanteen muutoksista, myös asiakkaan pilottiprojektit ovat taipuvaisia muutoksiin projektin edetessä tuotekehitysluonteestaan johtuen.

Huonosti määritellyt vaatimukset aiheuttavat väärinkäsityksiä toimittajan ja asiakkaan välillä. Vaatimusten ymmärrettävyyden vuoksi vaatimusten tulee olla [28]:

- tunnistettavissa yksilöllisillä identifikaatioilla
- selkeät sekä yksiselitteiset
- valmiit
- toteuttamiskelpoiset
- yhdenmukaiset
- testattavissa

Sovelluskehitysprojektilta vaaditaan muutostenhallintaa, jolla käsitellään vaatimuksiin tulevia muutoksia. Vaatimusten muutostenhallintaa varten tulee olla sovitut menetelmät, kuten millä menetelmällä asiakasvaatimuksiin muutoksia hyväksy-



Kuva 5.3 *Vaatimustenhallinnan CMMI:n mukainen prosessikaavio [28].*

tään. Muutosten ongelmana on niiden vaikutuksen yltäminen läpi ohjelmistoprojektin. Siksi eteenpäin ja taaksepäin jäljitettävyys ovat muutostenhallinnan kannalta tärkeitä. Jäljitettävyydellä katetaan katkeamaton ketju asiakasvaatimuksista aina toteutukseen asti. Sen avulla voidaan päätellä minne kaikkialle asiakasvaatimuksen muuttaminen tai poistaminen vaikuttaa toteutuksessa sekä mitä asiakasvaatimuksia jää puuttumaan, jos tietty osa järjestelmää on vielä toteuttamatta. Jäljitettävyyden ylläpitämisen katsotaan yleisesti olevan työlästä, mutta sitä edellytetään tiettyjen toimialojen standardoiduissa toimintatavoissa [12].

CMMI:n kypsyystasolla 2 vaatimustenhallinnalta vaaditaan tiettyjen käytänteiden suorittamista. Merkittävimpinä vaiheina ovat vaatimusten ymmärtäminen sekä sitoutuminen. Käsiteltäessä vaatimuksiin tulevia muutoksia, on muutostenhallinta keskeisessä osassa vaatimustenhallintaa. Tähän vaikuttaa ylläpidetty jäljitettävyys, jonka laiminlyönti hankaloittaa muutostenhallintaa. Kokonaiskuva CMMI:n mukaisesta vaatimustenhallinnasta on esitetty kuvan 5.3 prosessikaaviossa.

5.2.3 Vaatimusten jäljitettävyys

Jäljitettävyyden avulla pystytään osoittamaan yksittäisestä asiakasvaatimuksesta sen toteuttamiseen liittyvä ohjelmakoodi dokumentaation avulla. Jäljitettävyyttä käsitellään useissa ohjelmistosuunnittelua käsittelevissä standardeissa. Turvallisuuskriittisillä toimialoilla, kuten lääketeollisuus tai ilmailuala, sovellussuunnittelulta erityisesti vaaditaan jäljitettävyyttä. Tällöin jäljitettävyysanalyysillä voidaan todentaa, että toteutetussa sovelluksessa on implementoitu kaikki ohjelmistolle asete-

tut vaatimukset. Ohjelmakoodi on siis oltava yhdistettävissä siihen liittyvään spesifikaatioon sekä testaustoimenpiteisiin, eli jäljitettävyyttä käytetään esittämään vaatimusten ja toteutetun työn välisiä suhteita.

Kattavassa jäljitettävyydessä vaatimukset linkitetään toteutuksen suunnitteludokumenttiin, ohjelmakoodiin, sekä vaatimuksen todentavaan testitapaukseen. Tietyissä tapauksissa jäljitettävyydellä voidaan parantaa ohjelmakoodin uudelleenkäytettävyyttä. Mikäli eri projekteissa toistuvat samat vaatimukset, voidaan jäljitettävyysdokumentaation avulla valita sovelluskohteeseen jo toteutetut ohjelmakomponentit [5].

Jäljitettävyys esiintyy eteenpäin jäljitettävyytenä sekä taaksepäin jäljitettävyytenä. Eteenpäin jäljitettävyys mahdollistaa asiakkaan vaatimusmäärittelyn vaatimusten seuraamisen aina sen toteuttavaan ohjelmakoodiin saakka. Taaksepäin jäljitettävyydessä sen avulla edetään toiseen suuntaan, eli voidaan päätellä tuotetun ohjelmakoodin toteuttavan tietyn asiakasvaatimuksen. Jäljitettävyydellä on vaikutuksia myös muutostenhallintaan. Kun asiakas haluaa muutoksia vaatimuksiin, vaatimusten jäljitettävyydellä voidaan selvittää kyseisten muutosten vaikutuslaajuus toteutukseen. Tästä huolehtiva dokumentaatio vaatii ylläpitoa seuraavista seikoista: asiakasvaatimusten dokumentointi, asiakasvaatimusten ja järjestelmän väliset linkit sekä asiakasvaatimusten keskinäiset riippuvuudet toisiinsa [11].

5.3 Konfiguraationhallinta

Konfiguraationhallinta muodostuu tuoterakenteen monimutkaisuuden hallitsevista menettelytavoista. Konfiguraationhallinta muodostaa merkittävän osan tuotteenhallinnasta. Näiden menettelytapoihin kuuluvat ohjelmiston osakomponenttien hallinta, tuotteen eri konfiguraatioiden hallinta sekä muutosten hallittu tekeminen. Konfiguraationhallinnalla pidetään suunnitteluaineisto hyvin organisoituneena sekä tallennetaan tuotteen historiatietoja jäljitettävyyden mahdollistamiseksi [1, s. 126]. Sovelluksen kehityksen näkökulmasta tuotteenhallinta kokonaisuudessaan antaa sovellussuunnittelijoille hallitun ympäristön, jossa selkeillä pelisäännöillä saadaan aikaiseksi vakaa työskentely-ympäristö. Asiakastoimitusten kannalta tuotteenhallinnan tavoite on asiakastoimitusten konfiguraationhallinta, jotta tiedetään mikä kokoonpano tuotteesta asiakkaalla on käytössä. Laadunhallintajärjestelmä määrittelee tuotteenhallinnan. ISO 9001 edellyttää tuotteenhallinnan muodollisia menettelyjä, joiden mukaan on tiedettävä ohjelmistokomponenttien nykyinen versio ja tila sekä jokaisen tuotteessa esiintyvän komponenttien versio. On pystyttävä estämään samanaikainen päivittäminen (samanaikaisen kehittämisen ongelma) ja kirjattava jokainen muutosehdotus [37].

Sovellussuunnittelussa sen avulla hallitaan sovelluksen kokoonpanoa ohjelmistolle asetettujen vaatimusten täyttymiseksi sekä tähän liittyvää dokumentaatiota. Oh-

jelmistotuotetta myydään asiakkaille eri konfiguraatioina. Tuotteen versioitumisen lisäksi, erilaisia konfiguraatioita tuotteelle aiheutuu vaihtoehtoisten ohjelmistoympäristöjen käytöstä, vaihtuvat laitevalinnat, työkaluohjelmistojen eri versioiden käyttäminen tuotteen kehityksen aikana sekä asiakaskohtaisten kustomointien myötä [36].

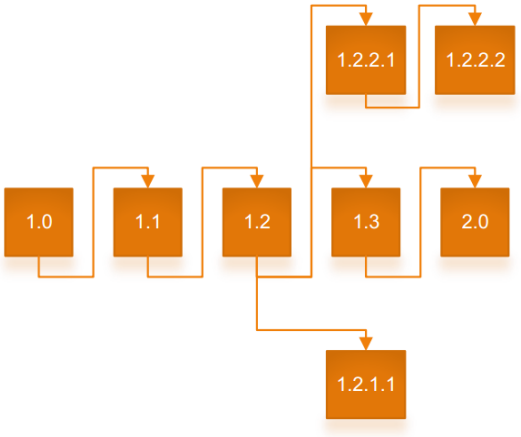
Sovellustoimitukset koostuvat osakokonaisuuksista, kuten ohjelman koodimoduulit, laitteet, toiminnallinen määrittely ja tukidokumentaatio. Nämä komponentit ja niiden versiot muuttuvat sovelluskehityksen aikana. Näistä voidaan koota eri konfiguraatioita eri tarkoituksiin. Tuotteen kehittyessä ajan kuluessa se siis versioituu. Tätä hallitaan versionhallinnan avulla jota käsitellään seuraavassa luvussa. Ohjelmistotuotteen normaalin versioitumisen lisäksi konfiguraatioihin vaikuttaa laitteiden ja kehitysympäristöjen variaatiot. Kehitysprojektin aikainen tuotteenhallinta poikkeaa ylläpidonaikaisesta. Ylläpitoon liittyy lisäksi analyysi mihin suunnitellut muutokset vaikuttavat, asiakaskohtaisten konfiguraatioiden selvittäminen sekä asiakkaalla käytössä olevien sovellusten kehitys- ja ajoympäristöjen versiotietojen hallinta [41].

Käytännön toteutus jäljitettävyyden linkeiksi vaatimusten ja toteutetun työn välillä tehdään vaatimusmäärittelyyn tarkoitetuilla työkaluilla tai käsin ylläpidettävällä jäljitettävyysmatriisi -dokumentilla. Ongelmaksi kattavan jäljitettävyyden luomisessa ja ylläpidettävyydessä muodostuu tarvittavien linkkien suuri määrä. Lisäksi laatuun liittyvien ei-toiminnallisten vaatimusten jäljitettävyys on hankala muodostaa, koska niiden vaikutus on laaja ja koko järjestelmän läpi leikkaava. Jäljitettävyyden avulla voidaan kuitenkin havaita toisensa pois sulkevia tai ristiriitaisia laatuvaatimuksia, sekä muodostaa parempi käsitys vaadituista laatutekijöistä. Mahdollisuudella yhdistää suunnittelun ja implementoinnin elementit toisiinsa, muodostuu parempi kuva jo toteutetuista ominaisuuksista ja niiden vaikutuksista [5].

Ylläpitovaiheessa konfiguraationhallinnan merkitys nousee esille, kun ohjelmiston ylläpidon aikana asiakkaalta tulee muutostoivomuksia ja raportoituja virheitä. Tällöin on tärkeää, että asiakaskohtaiset konfiguraatiot ovat tallennettu ja säilytetty. Tallentamisen kohteita eli konfiguraation muodostavia yksiköitä ovat ohjelmiston määrittely- ja suunnitteludokumentaatio, ohjelmiston lähdekoodit, testaussuunnitelmat ja raportit, todentamiseen ja kelpoistukseen liittyvät katselmoitavat dokumentit, valmiit ohjelmistoelementit ja -paketit sekä kaikki työkalut ja kehitysympäristöt [36].

5.3.1 Versionhallinta

Versionhallinta on tuotteenhallinnan keskeinen osa. Sen avulla sovelluskehityksen aikana suunnittelijoille mahdollistetaan hallittavissa oleva työympäristö. Kehitettävän sovellukseen muutokset tehdään aina versionhallinnan kautta, jotta estytään samanaikaisen kehittämisen ongelmalta. Ongelmassa on kyse toisen kehittäjän tekemien



Kuva 5.4 *Versioitumisen revisiot (peräkkäiset versiot) ja variaatiot (rinnakkaiset versiot).*

muutosten ylikirjoittamisesta. Versionhallinta joko estää samanaikaisten muutosten tekemisen, tai havaitsee ne ja pyrkii tietyin rajoittein yhdistämään molemmat muutokset kohteeseen. Lisäksi versionhallinta tarjoaa säilön, jonka avulla voidaan jakaa kehitetyn ohjelmiston versiot sovelluksen kehittäjien kesken sekä varastoida eri versioiden kopiot ylläpidon tarpeisiin.

Ohjelmistosovelluksien ja näiden komponenttien versioita numeroidaan usein kolmitasoisella numeroinnilla (esim. versio 1.2.3). Ensimmäinen numero vastaa kokonaan uudelleen kirjoitettua koodia. Toista numeroa käytetään uusia ominaisuuksia lisättäessä ja kolmatta numeroa käytetään virhekorjausten yhteydessä. Tuotekehityksessä käytössä voi olla neljäs numero, joka ei näy asiakkaalle. Peräkkäistä versiointia kutsutaan termillä revisio ja rinnakkaista versiointia kutsutaan variaatioksi, näiden rakentuminen on esitetty kuvassa 5.4.

Variaatio, eli versiopuun uusi haara, joudutaan tekemään tilanteessa jossa asiakkaan vanhaan versioon pitää lisätä uusia toimintoja tai korjata virhe, eikä asiakkaan ajoympäristön takia voida käyttää ohjelmakomponentin uusinta versiota. Mikäli toteutetut muutokset ovat sen luontoisia, että ne vaaditaan muihinkin sovellusversioihin, pitää ne lisätä myös uusimpiin versioihin ja näiden eri variaatioihin.

Sovelluksesta tehtävien uusien variaatioiden yhteydessä on huomioitava voidaanko sovelluksen uusimman version muutokset ottaa käyttöön sovelluksen suoritusaikana, vai vaaditaanko järjestelmän pysäyttäminen ja uudelleenkäynnistäminen muutosten lataamiseksi. Versiopuun haaroittamiseen vaaditaan hyvät perusteet, sillä haaroittuminen voi aiheuttaa ongelmia ylläpitäjälle. Versionhallinnassa huomioidaan myös laitesidonnaisuus [41].

Versiopuita hallitaan versionhallintaohjelmistojen avulla, joissa versioista ylläpi-

dettäviä tietoja ovat versionumeron lisäksi vastuuhenkilö, tila (valmis, tekeillä, ei aloitettu) ja tilan muuttumispäivämäärät. Lisäksi versionhallinnan piiriin kuuluu ohjelmakomponenttien dokumentaation versiointi.

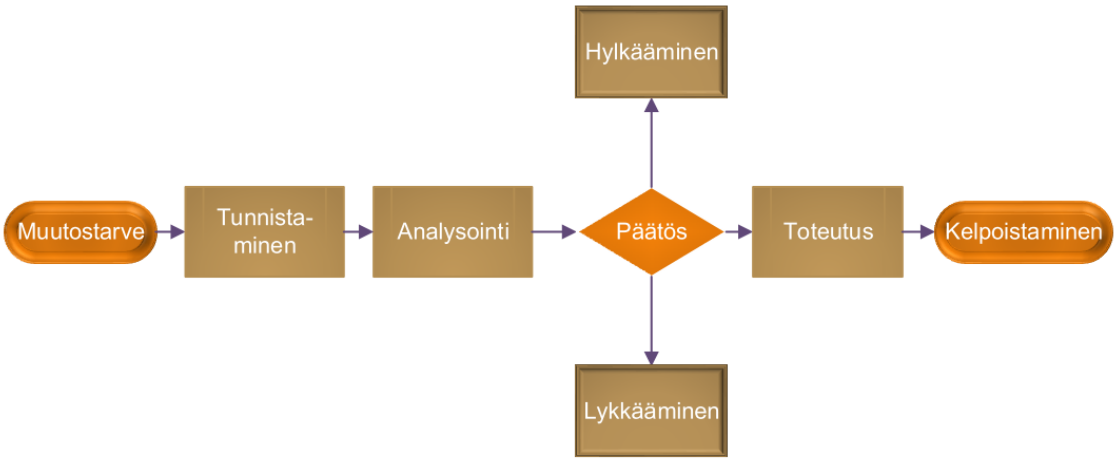
Toimivaa versionhallintaa tarvitaan, jotta virheiden korjaukset suuntautuvat oikeisiin ohjelmiston ja dokumentaation versioihin. Asiakkaan havaitsemien virheiden korjaamiseksi ja muutosten toteuttamiseksi on ylläpidon selvitettävä asiakkaan käytössä oleva versio tuotteesta. Jotta kaikkien tuotteen vanhojen versioiden virheet on tarvittaessa korjattavissa, pitää kaikki sovellusversiot olla säilössä. Näiden lisäksi kehitysympäristöjen ja työkaluohjelmistojen vanhat versiot olisi pidettävä mukana tuotteenhallinnan piirissä. Tällä varmistutaan siitä, että asiakkaan vanhoihin laiteympäristöihin yhteensopivat työkalut ja versiot tuotteesta ovat ylläpitoaikana saatavilla. Virheenkorjauksen jälkeen on selvitettävä mihin tuotteen muihin versioihin muutokset olisi syytä tehdä ja selvitettävä kuinka laaja vaikutus muutoksella on sovelluksen sisällä [41].

5.3.2 Muutostenhallinta

Muutostenhallinta kuuluu osaksi ohjelmistoprojektin tukitoimintoja ja voidaan luokitella osaksi tuotteenhallintaa. Muutostenhallinnassa voi olla kyse sovelluskehitysprojektin aikaisesta vaatimusmuutoksesta, jolloin se voidaan käsitellä vaatimustenhallinnalla, tai kyseessä voi olla ylläpidon aikaisten muutosten hallitseminen. Pienemmissä projekteissa projektinaikaiset muutokset voidaan hoitaa vaatimustenhallinnalla ja projektin päätyttyä siirrytään muutostenhallintaan. Ohjelmistotuotannon ketterissä pyrähdysmäisissä kehitysmalleissa muutostenhallinta on sisällytetty ohjelmistoprojektimalliin. Ketterässä kehitysmallissa uusia muutoksia ei huomioida kesken käynnissä olevan kehityspyrähdyksen, vaan ne otetaan mukaan vasta seuraavaan kehitysiteraatioon [11].

ISO 9001 standardissa mainitaan, että suunnittelun ja kehityksen muutoksista pidetään tallenteita. Standardin mukaan muutokset katselmoidaan, todennetaan ja kelpoistetaan. Ennen muutosten käyttöönottamista muutoksille hankitaan hyväksyntä. Muutosten vaikutusta tuotteen eri osiin, ja jo toimitettuun tuotteeseen analysoidaan sekä tarkastellaan järjestämällä katselmuksia.

Muutostenhallinta koostuu seuraavista päävaiheista: muutosten ehdottaminen, vaikutusten arviointi, päätöksen tekeminen, viestintä sekä vaatimusten vakiintumisen seuranta [48, s. 268]. Muutostarpeet tunnistetaan, arvioidaan ja priorisoidaan muutostenhallinnan avulla. Tämän tehtävänä on myös varmistaa muutosten vaatimustenmukaisuus. Sovelluksen muuttamiseen löytyy usein syyksi löytynyt virhe, toimintaympäristön aiheuttama muutospaine tai uuden ominaisuuden lisääminen. Tällaiset muutokset luokitellaan liiketoiminnallisiksi muutoksiksi, joiden avulla sovelluksen liiketoiminnallista hyötyä parannetaan [12].



Kuva 5.5 *Muutostenhallinnan yleinen prosessi [26].*

Teknisillä muutoksilla voidaan kehittää sovelluksen ei-toiminnallisia laatutekijöitä esimerkiksi parantamalla ylläpidettävyyttä. Tämän luontoisia muutoksia tehdään usein muiden muutostöiden lomassa. Tekniset muutokset eivät tuo sovellukselle liiketoiminnallista lisäarvoa, mutta kohentavat sovelluksen laatua. Sovelluksen ei-toiminnallisten ominaisuuksien parantamiseen kuitenkin vaikuttaa pääasiassa ulkoisesta vaatimuksesta tai niiden tuoman liiketoiminnallisen hyödyn takia [30].

Muutostarpeen synnyttyä muutostenhallintaa kuvassa 5.5 kuvatun prosessin ensimmäisenä vaiheena vastaanotetaan muutosehdotus. Ongelmatilanteessa paikannetaan ja tunnistetaan ongelma. Analysointivaiheessa arvioidaan muutoksen vaikutuksia onko muutoksen toteuttaminen kannattavaa sekä etsitään toteutusvaihtoehtoja. Näiden tulosten perusteella päätetään muutoksen toteuttamisesta. Jos muutos on toteutettavissa, toteutuksen jälkeen suoritetaan kelpoistaminen jonka avulla varmistutaan toteutuksen vastaavuudesta muutostarpeeseen. Muutos joka ei läpäise kelpuutusta palautuu aikaisempaan vaiheeseen, kunnes muutos hyväksytään tai vaihtoehtoisesti peruutetaan [26].

Ohjelmistoprojektin aikana tehtyjen katselmuksien, tarkastuksien sekä hyväksymisien jälkeen tehtävät muutokset dokumentoidaan sekä hyväksytään. Muutosten käsittelytavat sovitaan asiakkaan ja toimittajan välillä, jotta suunnitelmien jäädytyspäivän jälkeen tulevien muutosten tiedottaminen ja jakelu toimii ilman yllätyksiä. Muutostenhallinta on mukana ohjelmistotuotteen koko elinkaaren ajan. Vaativa muutos voidaan toteuttaa omana projektinaan. Projektissa muutos suunnitellaan tekemällä muutos- ja testaussuunnitelmat sekä kelpoistussuunnitelma. Suunnitelmien avulla kuvataan eri osa-alueiden työt. Toteutettujen muutosten myötä aloitetaan uusi kelpoistusjakso, jonka aikana tarkastellaan ainoastaan muutosten vaikutusten laajuutta [1].

elinkaarivaihe	dokumentit
Esiselvitys	ohjelmiston toiminnallisuuksien alustavat määrittelyt, liiketoimintasuunnitelma
Määrittely	käyttäjävaatimukset, sopimus, lähtötiedot (toimintakuvaus, I/O-luettelo, PI-kaavio jne.), alustava testaussuunnitelma
Suunnittelu	toteutuskuvaukset ja tukidokumentaatio: ohjelmistokuvaus, arkkitehtuurisuunnitelma, alustava käyttöohje, testaussuunnitelma ja testauslomakkeet
Toteutus	koodin dokumentointi, tukidokumentaation täydentäminen
Tehdastestaus	I/O-testauspöytäkirja, tehdastestauspöytäkirja
Käyttöönotto	käyttöopas, testauspöytäkirjat, mittauspöytäkirjat
Hyväksyntätestaus	hyväksyntätestaussuunnitelma
Käyttö ja ylläpito	käytön aikaisiin häiriötilanteisiin, virheisiin ja puutteisiin liittyvät raportit; käyttäjiltä saatu palaute, muutostoiveet ja kehitysehdotukset; asiakastyytyväisyyskysely; versionhallinnan, muutostenhallinnan ja konfiguraationhallinnan dokumentit; ylläpitosuunnitelma

Kuva 5.6 *Automaation sovellussuunnittelussa muodostettava dokumentaatio eri elinkaarivaiheissa.*

5.4 Dokumentointi

Sovelluskehityksen seurannan, projektinhallinnan, laadunvarmistuksen ja koko projektin onnistumisen kannalta asianmukainen dokumentaation on erittäin tärkeää. Dokumentointi parantaa ohjelmistokehityksen näkyvyyttä mahdollistaen projektin etenemisen seurannan. Dokumentointia hyödynnetään osana sovelluksen laadunvarmistusta myös tarkastusmenettelyjen muodossa. Näin varmistutaan siitä, että ohjelmisto vastaa käyttötarkoitustaan ja täyttää sille asetetut vaatimukset. Ohjelmistoprojektin dokumentaatio jää kuitenkin usein keskeneräiseksi projektikiireen tai epäsystemaattisten työmenetelmien takia. Puutteiden lisäksi ongelmia aiheuttaa muutosten päivittämättä jättäminen. Tämän seurauksena jo laaditut kattavatkin dokumentit vanhentuvat ja jäävät nopeasti hyödyttömiksi. Kuitenkaan kaikkea dokumentaatiota ei ole tarpeellista ylläpitää koko projektin ajan [22].

Sovellussuunnittelun elinkaaren eri vaiheissa muodostetaan dokumentaatiota vaiheen tapahtumista raporttien ja pöytäkirjojen muodossa sekä suunnitteludokumentaatiota lähtötiedoiksi seuraavaa vaihetta varten. Automaatiosovelluksen dokumentaatiota on listattu taulukkoon 5.6, jossa luetellaan sovellussuunnittelun päävaiheiden aikana muodostettavaa tukidokumentaatiota ja spesifikaatioita,

Sovelluskehityksen suunnitteluvaiheessa toteutettu arkkitehtuurikuvaus luokitellaan erääksi tärkeimmistä dokumenteista. Kuvauksessa esitetään suunnitellun toteutuksen perusrakenne ja filosofia. Lisäksi arkkitehtuurisuunnitelma toimii perehdytysmateriaalina toteutukseen jota voidaan hyödyntää ylläpidossa. Se auttaa hahmottamaan sovelluksen kokonaiskuvan. Arkkitehtuurisuunnittelua tehdessä keskitytään järjestelmän muuttumattomiin asioihin ja vastaavasti rajataan ulkopuolelle

helposti muuttuvat sekä kokonaisuuden kannalta turhat yksityiskohdat. Näillä rajauksilla dokumentti sopii paremmin ylläpitoa ajatellen eikä vanhene pienten muutosten myötä. Sovelluskehityksen toteutusvaiheen dokumentointina toimii kehitettävän sovelluksen kommentointi. Tämä tulisi suorittaa kuvaamalla tarkoitus ja ratkaisu abstraktilla tasolla eikä kommentoida ohjelmointikielen semantiikkaa. Tällöin ohjelmakoodi saadaan sisältämään toimintalogiikkaan liittyvä dokumentaatio [11].

6. DOKUMENTOITU SOVELLUSSUUNNITTELUPROSESSI

Osana tätä diplomityötä dokumentoitiin kohdeyrityksen sovellussuunnitteluprosessi. Jotta laadunvarmistuksen prosessia varten voidaan muodostaa kehitysehdotus, tulee ensin kartoittaa lähtötilanne. Tässä luvussa esitettävää sovellussuunnittelun prosessia kehitetään seuraavassa luvussa täydentämällä sitä laadunvarmistuksen menetelmillä. Dokumentointi suoritettiin asiantutijahaastatteluilla kootun aineiston perusteella. Tässä käsiteltäviä asioita ei aina välttämättä liitetä suoraan tiettyyn haastateltavaan, vaan viitataan haastatteluaineistoon. Erityisesti tapauksissa joissa asia on tullut esille useammassa haastattelussa. Haastateltavat asiantuntijat koostuivat kohdeyrityksen eri toimialasegmenttien kokeneista työntekijöistä.

6.1 Prosessin kuvaaminen

Jotta kaikkien eri organisaatioyksiköiden toiminta pystytään kuvaamaan saman prosessikaavion avulla, esitetään asiat sopivalta abstraktiotasolta. Asiantuntijahaastattelut suoritettiin yhden haastattelukierroksen aikana. Useammalle haastattelukierrokselle ei ollut tarvetta, koska yksittäisen haastattelun aikana saatu aineisto riitti ylemmän tason esityksen muodostamiseen.

Sovelluskehitystoiminta rakentuu elinkaarimallin ympärille. Elinkaarimalli voidaan jakaa perus ja tukitoimintoihin. Perustoiminnot muodostuvat sovelluksen kehitysprosessin vaiheista, kuten määrittely-, suunnittelu-, toteutus-, testaus- ja käyttöönotto. Liiketoimintaprosessi voidaan kuvata käyttämällä peräkkäisiä input-prosessi-output -kaavioita tai taulukkoesityksiä. Taulukkokuvausta käytettäessä prosessimalliin voidaan esimerkiksi liittää viittaukset laadunhallintajärjestelmän ohjeistukseen [42, s. 25].

Haastatteluaineistosta oli havaittavassa toimialasegmenttikohtaisia eroja. Tämä ilmeni siten, että kaikkien eri yksiköiden asiantuntijahaastatteluissa ei tullut ilmi jokaista tässä esitettävää vaiheitta tai tehtävää. Haastattelujen moninaiseen aineistoon vaikuttaa haastateltavien henkilökohtaiset tottumukset työn suorittamisesta, toimialakohtaiset käytänteet ja toimialalle tyypillisten projektien luonne. Siksi tässä esitettävä prosessikaavio on muodostettu kokoamalla haastatteluaineistosta esiin tulleet tehtävät. Kuvaus on toteutettu yleisellä tasolla toimialakohtaisten erojen takia, jotta dokumentoitu sovellussuunnittelun prosessi läpi leikkaa kaikkea yrityksen so-

vellussuunnittelua tämän työn kontekstissa. Yksityiskohtien esittäminen ei siis mahdollistaisi toimialariippumattoman esityksen luomista, joka on eräs työlle asetetuista vaatimuksista. Seuraavassa luvussa esitettävä dokumentoitu suunnitteluprosessi on siis yhteenveto haastattelujen aikana saadusta aineistosta.

6.2 Haastatteluaineiston mukainen sovellussuunnitteluprosessi

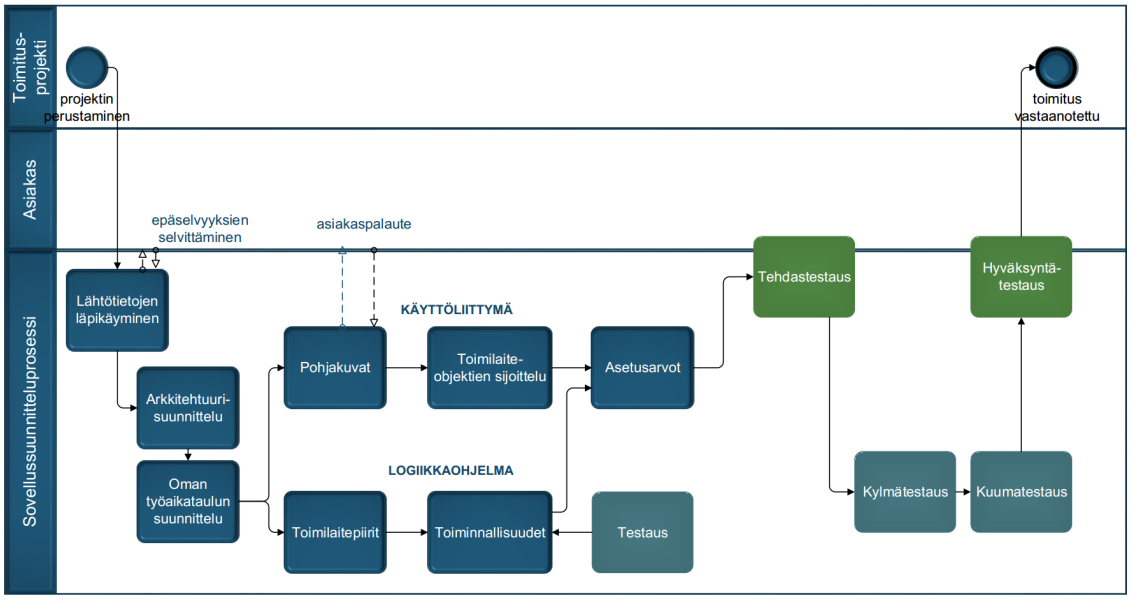
Automaation sovellussuunnitteluprosessi esitetään tässä jaoteltuna kahteen alalukuun, sovellussuunnitteluun sekä järjestelmän käyttöönottoon. Näistä ensimmäisessä käsitellään varsinaista sovellussuunnittelua joka koostuu työtehtävistä sovelluksen kehitystyön aikana. Jälkimmäisen osan muodostavat asiakkaan luona tapahtuva käyttöönotto ja siihen liittyvät sovellussuunnittelijan suorittamat toimet. Kovinkaan yksityiskohtaisiin detaljitietoihin ei tämän työn laajuus huomioiden ole mahdollista pureutua usean eri organisaatioyksikön kohdalla. Eikä välttämättä olisi mielekästäkään, koska jo pelkästään organisaatioyksikön sisällä projektien välillä on suuria eroja. Edellä mainitut kohdeyrityksen eri toimialoista vastaavat organisaatioyksiköt joiden asiantuntijoita haastateltiin, ovat seuraavat:

- energia- ja prosessiteollisuus
- ympäristö- ja vesihuolto
- elintarvike- ja kemianteollisuus
- koneet ja näyttämöt

Seuraavaksi esiteltävän kohdeyrityksen sovellussuunnittelun prosessin yleisen esityksen jälkeen pureudutaan organisaatioyksiköiden välisiin eroavaisuuksiin / samankaltaisuuksiin.

6.2.1 Sovellussuunnittelu

Kuvassa 6.1 esitetään kaavioesitys kohdeyrityksen sovellussuunnittelun prosessista joka on muodostettu haastatteluaineiston perusteella. Kuvan esitys keskittyy automaatiosovelluksen kehittämisen aikana tapahtuviin päätehtäviin, jotka sijoittuvat suunnittelijan suorittamista tehtävistä merkittävimmiksi. Prosessikaavio on siis toteutettu ainoastaan sovellussuunnittelijan näkökulmasta, eikä siinä oteta kantaa projektinjohdollisiin asioihin. Asiakkaan osallistuminen tiettyihin vaiheisiin esitetään sijoittamalla tehtäviä kuvaavat lohkot osin myös asiakkaan vastuualueelle. Seuraavassa avataan tehtävien sisältöä. Tehtävien kuvaukset on jaoteltu ohjelmistokehityksen päävaiheiden mukaisesti otsikoituihin kappaleisiin: määrittely, suunnittelu, toteutusvaihe (toiminnallisuudet ja käyttöliittymä), tehdastestaus sekä käyttöönotto.



Kuva 6.1 *Asiantuntijahaastattelujen avulla dokumentoitu kohdeyrityksen sovellussuunnitteluprosessi.*

Määrittely

Sovellussuunnittelijan työtehtävät alkavat usein jo määrittelyvaiheessa. Haastatteluaineiston mukaan ensimmäiseksi suoritetaan *lähtötietojen läpikäyminen*. Koska mikään ei korvaa lähtötiedoissa kuvatun järjestelmän ymmärtämistä, ensimmäisenä tehtävänä suunnittelijalla on selvittää automaatioon liitettävät laitteet ja ymmärtää kohdeprosessin toiminnan periaate. Mikäli näistä laitteista ei ole saatavilla kattavaa I/O-luetteloa tulee suunnittelijan muodostaa se itse olemassa olevasta dokumentaation avulla sekä asiakasta haastattelemalla [16]. Sovellussuunnittelijan tehtäviin voi kuulua myös muita määrittelyvaiheen asioita, kuten toimintakuvauksen muodostaminen tai säätökaavion suunnittelu kohdeprosessia varten [24]. Määrittelyn aikana käydään ajotapakeskustelua asiakkaan kanssa ja muodostetaan lähtötietoja täydentävä dokumentaatio jonka pohjalta toteutettava järjestelmä suunnitellaan. Ajotapa kuvaa miten järjestelmä toteutetaan. Tämä hyväksytetään asiakkaalla ja hyväksynnän jälkeen tulevat muutokset jäädytettyihin lähtötietoihin nähdään lisäkustannuksina [19].

Suunnittelu

Määrittelyvaiheen jälkeen automaatioprojektin lähtötiedot tulee olla mahdollisimman hyvin selvitetty ennen seuraavaan vaiheeseen siirtymistä. Suunnitteluvaihe muodostuu tässä dokumentoidussa suunnitteluprosessissa ainoastaan sovelluksen suunnittelusta sekä toteutuksen aikataulun muodostamisesta. Tämän takia mahdollinen muu suunnittelutyö ajatellaan kuuluvan sovelluksen määrittelyvaiheeseen, jossa

määriteltiin kaikki ne tekijät joiden mukaan sovelluksen tulee toimia. Sovellussuunnittelijan tehtävänä suunnitteluvaiheessa on muodostaa käsitys sovelluksen rakenteesta. Kun automaatioon liitettävät laitteet ja kohdeprosessin ajotapa on selvillä, voidaan sovelluksen *arkkitehtuurisuunnitelu* suorittaa.

Tämän jälkeen kun laitteisto ja toiminnallisuudet ovat selvillä sekä sovelluksen arkkitehtuuri hahmoteltu, pystyy suunnittelija muodostamaan *oman työaikataulun suunnittelun* jonka perusteella hän toteuttaa sovelluksen kehittämisen. Oman työaikataulun mukaan suunnittelijalla on mahdollisuus seurata työnsä edistymistä. Mikäli jokin aikataulutettu osa-vaihe jää jälkeen, on tämä havaittavissa mahdollisimman aikaisessa vaiheessa. Tällöin suunnittelija pystyy kohdentamaan omaa ajankäyttöänsä oikeisiin asioihin tai pyytämään lisäapua työn suorittamiseen. Tällä turvataan projektin pysyminen aikataulussa suunnittelun osalta [16].

Suunnittelun yhteydessä muodostuu myös kuva toteutuksen hankalimmista osa-alueista, joiden toteuttaminen vaatii tavallista suuremman työpanoksen. Suunnittelutilanteessa huomioitavia hankalia osa-alueita muodostuu esimerkiksi uuden teknologian hyödyntämisestä, tuotekehitystä vaativista osuuksista, ulkoisten järjestelmien liittämisestä tai vaatimusmäärittelyn epäselvyyksistä. Automatisoitavaan kohdeprosessiin perehtyminen jatkuu pitkin suunnitteluvaihetta. Määrittelyvaiheessa aloitettua ajotapakeskustelua jatketaan asiakkaan kanssa ja ajotapakeskustelut jatkuvatkin usein koko projektin ajan.

Toteutusvaihe, automaatiosovellus

Toteutus logiikkaohjelmoinnin osalta alkaa muodostamalla kehitystyökaluun laitteistokonfiguraatio sekä *toimilaitepiirit* mittauksia, moottoreita, venttiileitä, säätimiä ja digitaalituloja ja -ohjauksia varten, eli kaikkia niitä toimilaitteita varten jotka kytketään automaatioon. Toimilaitepiirien muodostamiseen hyödynnetään koodin uudelleenkäyttöä käyttämällä tuotekehityksen muodostamia toimilaitelohkoja. Pienemmissä logiikoissa on ainoastaan auki kirjoitettua koodia, joten niiden osalta ei voida toteuttaa piirejä [19]. Toimilaitepiirien muodostamiseen on tietyissä kehitysympäristöissä automatisoituja toimintoja. Näiden lisäksi logiikkaohjelmaan voidaan generoida rajapinnat käyttöliittymää ja ylemmän tason automaatiosovelluksia varten, mikäli näihin on kehitetty generointityökalut. Generointityökalujan hyödyntäminen suunnittelutyössä on siis toteutusalustariippuvaista.

Toimilaitepiirejä varten kehitetyt moduulit toteuttavat asiakkaiden vaatimia sekä yleisesti hyväksi havaittuja toimintoja. Eräitä tällaisia toimintoja ovat raportointidatan laskemista suorittavat moduulit, väylälaitteen kanssa kommunikointia suorittavat ajurilohkot sekä erilaisien lukituksien ja hälytyksien muodostamiseksi kehitetyt moduulilohkot. Mikäli uusia vastaavia moduuleja tarvitaan, tulee ne sovellussuunnittelijan kehittää itse.

Sovelluksen toiminnallisuudet on määritelty lähtötietoina saaduissa toimintakuvauksissa sekä määrittelyn aikana käytyjen ajotapakeskustelujen kautta. Määrittelyvaiheen lopputuloksena saatavan, luonnollisella kielellä kuvatun toimintakuvauksen perusteella toteutetaan *toiminnallisuudet*. Näiden suunnitelmien perusteella automaatiosovelluksen toiminnallisuudet ohjelmoidaan toteutusvaiheessa. Ne toteutetaan liittämällä laiterajapinnan toimilaitepiirit lähtötietojen mukaiseen toiminnallisuuteen. Ohjelmoinnin aikana pyritään muodostamaan mahdollisuuksien mukaan uudelleen käytettäväksi kelpaavia ohjelmamoduuleja. Uudelleenkäytettävyys huomioidaan esimerkiksi nimeämällä ohjelmamoduulin rajapinnan alkiot ja koodin kommentit sovelluskohdeneutraalisti [23]. Tällöin toiminnallisuuden ollessa hyvin dokumentoitua ei moduulia uudelleen käytettäessä toiminnallisuuden myöhemmin selvittämiseen tarvita projektin lähtötietoja, josta koodi on alunperin lähtöisin.

Testaus on sovelluskehityksen aikana tapahtuva luonnollinen keino varmistua tuotetun koodin ajonaikaisesta oikeellisuudesta. Kyseessä on ohjelmoinnissa lähes aina mukana oleva laadunvarmistustoimi. Aina kun sovelluksen osakokonaisuuksia saadaan valmiiksi, ne testataan. Näin varmistutaan vaatimusmäärittelyn mukaisuudesta mahdollisimman varhaisessa vaiheessa. Testaus suoritetaan jokaisen ohjelmamoduulin kehityksen aikana sekä moduuleja sovellukseksi integroitaessa. Tällöin kyseessä on *moduuli- ja integrointitestaus*. Lopullisen automaatiototeutuksen testausympäristön avulla suoritetaan tehdastestaus (FAT), jossa mukana on mahdollisuuksien mukaan automaatiokomponentteja ohjelmoitavan logiikan CPU:n lisäksi I/O-kortteja ja väylälaitteita, käyttöliittymänä valvomo ja/tai paneeli, väylässä oleva hajautus I/O, sekä muita väylään liitettäviä laitteita kuten taajuusmuuttaja tai toinen ohjelmoitava logiikka.

Toteutusvaihe, käyttöliittymä

Automaatiojärjestelmän käyttöliittymää voidaan ryhtyä kehittämään heti kun lähdemateriaalia on saatavilla. Prosessiteollisuudessa hyvänä lähtötietona on kohdeprosessia kuvaava prosessi-instrumentointikaavio. Työ aloitetaan graafisella suunnittelulla jossa kuvattu prosessi jaotellaan selkeiksi näyttökuviksi ja piirretään käyttöliittymän kehitystyökalulla. Käytettävänä ovat aikaisemmin vastaaviin projekteihin toteutetut graafiset elementit sekä kyseiselle kehitysympäristölle toteutetut toimilaiteobjektit, jotka suorittavat tiedonvaihdon logiikkaohjelman kanssa määritellyn rajapinnan avulla. Tässä vaiheessa ei välttämättä vielä tarvitse olla toimilaitepiirejä valmiina, mutta niiden graafinen huomioiminen alusta asti on onnistuneen käyttöliittymän edellytys.

Ensin voidaan hahmotella käyttöliittymän *pohjakuvat*. Parhaan tuloksen saa kun hahmottelee asiakkaan ja/tai loppuasiakkaan kanssa yhteistyössä prosessin sivujaon. Varsinainen käyttöliittymän piirto sekä *toimilaiteobjektien sijoittelu* suositellaan teh-

tävän samalla kertaa parhaan lopputuloksen aikaansaamiseksi, niin työn tuottavuuden kuin graafisen ulkoasun kannalta [16].

Käyttöliittymän kautta välittyy asiakkaalle subjektiivinen laatuvaikutelma toimitetusta järjestelmästä. Tämä muodostuu käyttöliittymän selkeydestä sekä informatiivisuudesta, sekä käyttöliittymän helppokäyttöisyydestä. Siinä vaiheessa kun valmiita käyttöliittymän pohjakuvia saadaan aikaiseksi, kysytään asiakkaalta näistä palautetta mahdollisimman aikaisessa vaiheessa [20]. Ei-toiminnalliseen laatuun voidaan käyttöliittymän osalta vaikuttaa esimerkiksi laajennettavuuden avulla. Laajennettavuus esiintyy käyttöliittymissä esitettävien tagien laajennettavuudella lisenssikäytänteiden avulla.

Käyttöliittymän pitää olla yhdenmukainen loppuasiakkaalla jo olevan valvomon osalta niissä tilanteissa, joissa jo olemassa olevaan valvomoon suunnitellaan uutta. Lisäksi suunnittelijoiden toteuttama graafinen esitys tulee olla yhdenmukaista keskenään, jos käyttöliittymä toteutetaan useiden suunnittelijoiden voimin [16],[23]. Asiakkaalla voi olla jo valvomon lähtötiedoissa vaatimuksena tietynlaisen prosessilaitteiden jaon määrittely tai ohjeet värien käytöstä. Mikäli asiakas ei ota kantaa näihin, käytetään hyvien käytänteiden mukaista esitystä ja kysytään niistä palautetta.

Automaatiosovelluksen toiminnallisuuksien toteutuksen edetessä prosessia ohjaavien ohjelmamoduulien syötteinä vaadittavien parametrien määrä kasvaa. Nämä toiminnallisuuksien parametrit sijoitetaan käyttöliittymään asetusarvoiksi. *Asetusarvot* voivat olla kerättynä erillisille valvomon sivuille sekä popup-ikkunoihin prosessilaitekokonaisuuksien mukaisesti, tai sijoitettuna suoraan prosessia kuvaavan grafiikan sekaan prosessilaitteiden läheisyyteen. Vasta kun kaikki toiminnallisuudet ovat ohjelmoitu ja testattu, tiedetään kaikki vaadittavat asetusarvot. Valvomorajapintaan on hyvä kerätä parametrit kohdeprosessin osa-aluekohtaisesti sekä huomioida mahdolliset muutokset ottamalla varalle tageja samalle muistialueelle jossa on muitakin samaan kokonaisuuteen liittyviä asetusarvoja. Kun käyttöliittymän prosessi- ja asetusarvografiikat ovat toteutettu, voidaan käyttöopas viimeistellä käyttöliittymän varsinaisien grafiikoiden kuvilla.

Tehdastestaus

Toteutusvaiheen päätteeksi tehdastestaus (FAT) tilaisuudessa esitellään asiakkaalle automaatiojärjestelmää sekä käyttöliittymää. Kenttälaitteiden herätteitä on mahdollista simuloida toimilaitepiireihin kehitetyillä ominaisuuksilla. Jos testattavana on myös sähköautomaatiokeskus voidaan herätteet antaa riviliitinrajapinnasta. Usein varsinaisten toiminnallisuuksien ja sekvenssien esittely tällä tavalla on hankalaa, koska simuloitavat piirit voivat vaatia useiden signaalien samanaikaista simulointia.

Ennen tehdastestausta projektia varten valmistetut sähköautomaatiokeskukset ja

hajautuskotelot testataan riviliitinrajapinnasta. Mikäli keskukset toimitetaan asennettavaksi ennen tehdastestausta, tulee ne tarkastaa aikaisemmin. Keskuksen testaaminen I/O-rajapinnan osalta kuuluu sovellussuunnittelijan työtehtäviin. Tämän tarkoituksena on varmistua siitä, että käyttöönotossa ilmenevät kytkentävirheet rajoittuvat sähkökeskuksen ulkopuolelle, joka nopeuttaa käyttöönottovaiheessa suoritettavaa virheidenetsintää [19]. Tässä testausvaiheessa usein yhdistetään automaatioväylä ensimmäistä kertaa väylälaitteiden välille. Testauksessa voi olla myös kyse oppimistapahtumasta jossa kokeneempi suunnittelija esittelee kyseisen teknologian järjestelmän ja väylän "pystyttämisen". Kaikki testattavat automaatiolaiteet parametroidaan järjestelmän testausympäristön perustamisen yhteydessä. I/O-testauksen yhteydessä kirjataan pöytäkirja josta tulisi ilmetä testauksen ajankohta, testauksen suorittaja ja testatut rajapinnat.

Mikäli sähköautomaatiokeskusta ei testata ennen asennusta, sovellussuunnittelija koestaa ainakin toimitettavat automaatiokomponentit [13]. Näin suunnittelija varmistuu automaatiolaitteiston toiminnasta ja nopeuttaa käyttöönottoa tekemällä osan työstä etukäteen. Jos projekti sisältää tuotekehitystä, on varsinaisilla laitteilla suoritettu testaaminen välttämätöntä. Yksi tärkeä huomioitava asia on myös se, että kehitetty sovellus on mahdollista ladata projektille hankittuun ohjelmoitavaan logiikkaan. Automaatiokomponenteissa käytettävä muisti vaikuttaa sen hankintahintaan. Vaikka muistin määrä on usein laajennettavissa muistikortin muodossa, on laitteiston eri mallien kesken olemassa muitakin rajoitteita, kuten datalohkojen maksimimäärä sekä väylälaitteiden kanssa kommunikointiin tarjolla olevien muistialueiden laajuus.

Laitteen resurssien kuluttamiseen voidaan vaikuttaa käytetyillä koodauskäytänteillä tai valitulla kommunikointiprotokollalla. Esimerkiksi valmiiden toimilaitepiirien yhteydessä niiden kuluttamat datalohkot rajoittavat kytkettävien toimilaitteiden määrää. Yleensä toiminnallisuuksia ohjelmoitaessa suunnittelijalta loppuu ensimmäisenä sähkökatkon yli säilyvät muistipaikat. Resurssien kulutus pitää siis tiedostaa jo laitehankintoja tehdessä, mutta usein joudutaan kulkemaan erilaisia kiertoteitä sovellukseen tarvittavien toimintojen toteuttamiseksi laitekustannusten rajoittamiseksi.

Tehdastestaus mahdollistaa asiakkaalle konkreettisen kokemuksen tulevan automaatiojärjestelmän operoimisesta. Samalla mahdollistuu muutosehdotuksien esittäminen käyttöliittymään ja toiminnallisuuksiin. Muutokset ehditään toteuttamaan vielä projektin aikana ennen käyttöönoton alkamista [19]. Tehdastestauksen laajuus perustuu asiakkaan vaatimuksiin. Tehdastestaukseksi voi riittää pelkän valvomon kautta toteutettu käyttöliittymän ja järjestelmän esittely, tai asiakas voi halutessaan tarkastella toiminnallisuuksia toteuttavaa koodia osana tehdastestausta [16]. Testauksen tavoitteena on saada asiakkaalta viimeiset muutokset ja hyväksyntä to-

teutukselle ennen käyttöönottoon siirtymistä. Tehdastesti päättyy hyväksyttyyn testauspöytäkirjaan. Mikäli tehdastesti päättyy hyväksyntään määritellyin muutoksin, ei uutta tehdastestiä tarvitse järjestää muutosten hyväksymiseksi [23]. Muutosten suorittamisen jälkeen toteutusvaihe päättyy ja sovellussuunnittelija valmistautuu käyttöönottoon siirtymiseen.

6.2.2 Käyttöönotto

Tämän luvun sisältö käsittää lyhyen kuvauksen käyttöönottovaiheen sekä hyväksyntätestauksen sisältämistä tehtävistä, jotka sovellussuunnittelijalla on edessä järjestelmän tuotannolle saattamisessa. Käyttöönottoon siirrytään tehdastestauksen jälkeen kun kohdeprosessin mekaniikka- ja sähköasennukset ovat siinä vaiheessa, että ensimmäisiä piirejä voidaan alkaa testaamaan.

Kylmätestaus

Käyttöönotto voidaan aloittaa, kun kohdeprosessin mekaniikka, instrumentointi- sekä sähkö- ja automaatioasennukset ovat saavuttaneet vaaditun valmiustason. Ensimmäisenä tehtävänä sähköistetään automaatiolaitteet ja pystytetään mahdollinen väylä sekä asetetaan parameterit laitteisiin ja sovellukseen. Tämän jälkeen voidaan aloittaa instrumentoinnin, moottorien ja venttiilien testaus.

Aikaisemmin suoritettavassa sähköautomaatiokeskuksen I/O-testauksessa voitiin varmistua siitä, että keskuksen riviliitinrajapintaan asti se toimii virheettä. Käyttöönoton ensimmäisenä päävaiheena varmistutaan kohdeprosessin kenttälaitteiden oikeasta toiminnasta. Myös tästä testauksesta tehdään pöytäkirja, jonka avulla pidetään kirjaa testatuista laitteista sekä varmistutaan, että jokainen automaatioon kytketty laite on testattu ja toimii oikein.

Koeajon aikana testataan, että automaatiosovellus ohjaa prosessilaitteita määrittelyn mukaisesti. Tämä tapahtuu suorittamalla sovelluksen toimintoja osakokonaisuus kerrallaan. Kylmätestaukseksi kutsutaan koeajoa, jossa prosessissa ei käytetä tuotannon aikaisia aineita tai kuormia, vaan ohjaukset ja sekvenssit testataan ohjaamalla tyhjiä laitteita tai vesiprosessia. Säätöpiirien alustavat viritykset voidaan suorittaa tässä vaiheessa, mutta säädöt viritetään varsinaisilla prosessikemikaaleilla seuraavassa vaiheessa.

Kuumatestaus

Kylmätestauksen jälkeen voidaan aloittaa kuumatestaus, jossa kohdeprosessia ajetaan kuormituksella tai varsinaisilla prosessiaineilla. Tällöin tavoitteena on saattaa kohdeprosessi tuotantokäyttöä vastaavaan tilaan. Testauksen aikana eri toiminnallisuuksia pyritään asettamaan automaattitoiminnolle sekä valitsemaan prosessin ajamiseen soveltuvat asetusarvot ja viritetään säätöpiirit. Asiakas ja loppuasiakas voi

olla mukana kuumatestauksissa.

Mikäli erillistä hyväksyntätestausta ei suoriteta, voidaan automaatiojärjestelmä hyväksyä asiakkaan toimesta jo kuumatestauksen avulla. Tällöin asiakkaan edustaja kuittaa allekirjoituksellaan jokaisen yksitellen testatun toiminnallisuuden, jonka seurauksena automatisoitu kohdeprosessi voidaan ottaa tuotantokäyttöön [20],[24]. Tuotantoon siirtymiseksi asiakkaan kanssa on kirjoitettu toiminnallisuuspöytäkirja jolla varmistetaan, että asiakkaan näkemys prosessin toiminnallisuudesta vastaa automaation toteuttamaa toiminnallisuutta.

Hyväksyntätestaus

Hyväksyntätestaus (SAT) voidaan toteuttaa ajamalla testattavaa prosessia toimintakuvauksen jokainen osakokonaisuus kerrallaan. Hyväksyntätestauksen suorittaminen riippuu asiakkaasta, eikä sitä välttämättä aina pidetä [19]. Testauksen tulokset kirjataan ylös ja automaatiototeutus katsotaan hyväksytyksi, kun jokainen toimintakuvauksen osa on hyväksytty [24]. Hyväksyntätestauksen ehdoksi voidaan asettaa myös tuotantoprosessin virheetön koekäyttö eli kelpoistusjakso, jonka aikana parametri- tai ohjelmamuutosten tekeminen ei ole sallittua. Koekäyttö on hyväksytty kun sen aikana on tapahtunut sopimuksen mukainen määrä määritellyn tyyppisiä virheitä [20]. Onnistuneen koekäytön jälkeen asiakas vastaanottaa toimituksen ja työaikainen vakuus loppuu ja siirrytään takuuaikaisen vakuuden piiriin [19].

Koulutus

Käyttöönottoon sisältyy myös asiakkaan kouluttaminen uuden järjestelmän käyttöön. Tämän suorittaa järjestelmän käyttöönottava sovellussuunnittelija. Mikäli järjestelmää varten on toteutettu käyttöopas, on tätä hyvä käyttää koulutuksen sisällön runkona. Koulutuksessa tulisi keskittyä prosessin operoimiseen ja jättää toteutettujen valikoiden sekä teknologian yksityiskohtainen esittely taka-alalle, vaikka tämä voi järjestelmän suunnitelleesta insinööristä tuntua vieraalta. Hyvä menetelmä operoinnin kouluttamiseen on valmistella tapauksia (case) operaattoreiden suoritettavaksi.

6.3 Organisaation segmenttikohtaiset suunnitteluprosessin vaiheet

Asiantuntijahaastattelujen avulla kerätystä aineistosta oli havaittavissa eroavaisuuksia eri toimialojen kesken. Näiden osalta sovellussuunnittelun työprosessit olivat muodostuneet ajan saatossa omaan muotoonsa johtuen osin asiakaskohtaisista eroavaisuuksista tai totutuista käytänteistä. Haastatteluiden aikana muodostui käsitys, että projektit etenevät eri toimialoilla eroavaisuuksista huolimatta systemaattisesti. Projektien suorittamisen luonteeseen vaikuttaa toimialakohtaiset vaatimukset

sekä hyödynnettävänä olevat kehitystyökalut. Kun aikaisempaa projektia vastaava uusi projekti suoritetaan, tapahtuu sovelluskehitys toistamalla aikaisempi onnistunut suoritus. Tätä toimintatapatulkintaa tukee esitetty vertaus, että kohdeyritys ISO 9001 sertifioituna organisaationa toimii CMMI kypsyysmallin mukaisella tasolla 2 [11, s. 148]. Tällöin kyseessä oli *toistettava* prosessi (luku 2.2 Prosessin kehittäminen).

Haastatteluaineistossa toistui maininta projektien työprosessissa esiintyvien eroavaisuuksien olevan projektikohtaista. Erityisesti tähän vaikuttavat asiakkaan/loppuasiakkaan vaatimukset. Eroavaisuudet esiintyivät erityisesti määrittelyvaiheen projektikohtaisina eroavaisuuksina sekä vaihtelevina asiakkaan kelpoistusvaatimusina.

Toimialasegmenttien sisällä määrittelyvaiheessa projektikohtaiset eroavaisuudet muodostuivat valmiina saatavien lähtötietojen määrässä ja laadussa. Määrittelyvaihe voi olla suoritettu kolmannen osapuolen toimesta, tai siihen vaaditaan sovellussuunnittelijalta työpanosta. Määrittelyvaiheesta seuraavana oleva elinkaarimallien suunnitteluvaihe jäi uupumaan kokonaan muutaman toimialasegmentin haastatteluaineistosta. Tästä voi välittyä kuva työprosesseihin muodostuneesta käytännöstä suoraan toteutukseen siirtymisestä hyppäämällä yli suunnitteluvaiheen. Tämä viestii automaation sovellussuunnittelun suoraviivaisuudesta johon vaikuttavat toteutusvaiheessa hyödynnettävä koodin uudelleenkäyttö, kehitystyökalut, haastateltavien asiantuntijoiden totutut työtavat sekä vuosien saatossa saavutettu perehtyneisyys toimialan kohdeprosesseihin. Automaatiosovelluksen toteutusta edeltää aina jonkinnäköinen suunnittelu, mutta tätä ei ole haastatteluaineiston perusteella totuttu tiedostamaan omaksi vaiheekseen, saati dokumentoimaan.

6.3.1 Elintarvike- ja kemianteollisuus

Elintarvike- ja kemianteollisuuden asiantuntijahaastattelun mukaan toteutusvaiheessa automaatiosovellus koodataan toimintakuvausten mukaisesti, niiltä osin kun epäselvyyksiä ei esiinny. Aivan kuten muillakin toimialoilla aluksi tarkistetaan lähtötietojen dokumentaatio ja selvitetään epäselvät kohdat. Toimilaitepiirit muodostetaan valmiista aikaisemmin kehitetyistä moduuleista. Tällä toimialalla paljon käytettävä kehitysympäristö tarjoaa myös valmiita moduuleja tätä varten, vaikuttaa omalta osaltaan tässä esitettäviin muista poikkeaviin käytänteisiin. Toiminnallisuuksien ohjelmoinnissa toteutettu sekvenssi testataan simuloimalla ennen kuin tuotettua koodia monistetaan ja käytetään uudelleen. Vastaavan tyyliset toimintatavat esiintyvät kaikkien kohdeyrityksen organisaatioyksiköiden eri segmenteillä.

Elintarvikealalla korostui toteutusvaiheen aluksi tapahtuva prosessin toimilaitepiirien tehdastestaus. Tämän tarkoituksena on testata käyttöliittymätoteutuksen toiminta sekä hyväksyttää asiakkaalla toimilaitteiden graafinen esitys (faceplate), ennen kuin näitä monistetaan satoja käyttöliittymäkehityksen aikana. Varsinaises-

sa tehdastestauksessa käydään läpi asiakkaan kanssa joko kaikki valvomokuvat ja ohjelman toiminta toimintakuvauksen tai lähtötietona saadun ohjelmistokuvauksen mukaisesti, tai asiakas valitsee pistokoeluontoisesti testattavat osa-alueet. Sähköautomaatiokeskuksen I/O-testaus suoritetaan, mikäli se kuuluu osaksi kohdeyrityksen automaatiotoimitusta.

Elintarviketeollisuudessa kylmätestaus pitää sisällään vesikoeajot. Testauspöytäkirjana käytetään usein lähtötietojen dokumentaatiota. Hyväksyntätestaus suoritetaan myös toimintakuvauksen mukaisesti, jokainen sovelluksen piiri testataan hyväksyvän osapuolen kanssa yhdessä. Elintarviketeollisuudessa voidaan suorittaa automaation- ja koko prosessin kelpoistusjaksoja tapauskohtaisen sopimuksen määrittelemällä tavalla. Kelpoistusjakso esiintyy dokumentoidussa kohdeyrityksen sovellussuunnittelun prosessissa, vaikka on täysin projektikohtaista vaaditaanko sitä suoritettavaksi.

6.3.2 Koneistot ja näyttämöt

Koneistot ja näyttämöt toimialasegmentin osalta seuraavaksi käydään pintapuolisesti näyttämötekniikkaprojektin työvaiheet läpi. Näyttämö- ja teatteriautomaatiossa mukana on SIL3 -tason TLJ. Näyttämötekniikan projekteihin liittyy aina laadunvarmistussuunnitelma sekä riskianalyysi, mutta turvaohjelman laadunvarmistusta ei ole totuttu suorittamaan energiatekniikan tapauksessa esitellyllä menettelyllä.

Sovellussuunnitteluprosessin määrittelyvaiheeseen kuuluu haastatteluaineiston mukainen lähtötietojen läpikäyminen sekä näissä esiintyvien epäjohdonmukaisuuksien ja puutteiden selvittäminen. Tästä siirrytään toteutusvaiheeseen, jossa käyttöautomaation osuudessa voidaan hyödyntää tähän kehitettyjä generointityökaluja. Turvaohjelma ohjelmoidaan käsityönä, koska vaadittava turvaohjelma on aina tapauskohtainen. TLJ-sovellus muodostaa lisävaiheen myös toteutusvaiheen integrointiosuuteen, jossa se integroidaan käyttöautomaatio-ohjelman kanssa.

Näyttämötekniikassa käyttöliittymä muodostuu joko paneelista tai valvomosta ja paneelista. Ohjelmoimiseen käytetään mahdollisimman paljon aikaisempien projektien aikana toteutettua työtä, mutta pääasiassa työt toteutetaan käsin. Näyttämötekniikkaprojekteissa ei pääsääntöisesti suoriteta tehdastestausta. Tämän seurauksena käyttöönoton I/O-testauksessa havaitut virheet pitää tarkastaa myös sähköautomaatiokeskuksen osalta, joka lisää käyttöönoton työmäärää.

Käyttöönoton aikaisesta työstä ison osan muodostavat liikutettavien nostimien ääriasentorajojen mekaaninen sekä sähköinen virittäminen. Käyttöönotto alkaa yleiseen tapaan automaatiolaitteiston ja väylän pystyttämisellä sekä kylmätestaukseen kuuluvalla I/O-testillä. Kun ajoparametrit ovat aseteltu automaatiolaitteisiin ja testaus suoritettu, siirrytään suorittamaan koeajoa. Kylmätestauksella on samanlainen merkitys kuin muilla toimialoilla. Sen aikana tarkastetaan ohjausohjelmiston sekä

mekaniikan toiminta. Ohjaukset suoritetaan jokaisesta käyttöliittymälaitteesta. Testaaminen toteutetaan suunnitelman mukaisesti ja tulokset kirjataan pöytäkirjaan. Tässä vaiheessa TLJ asetetaan turvaohituksilla pois käytöstä kylmätestauksen ajaksi.

Näyttämötekniikan käyttöönotossa suoritettavat työvaiheet sisältävät antureiden kalibrointia ja virittämistä. Näihin lukeutuvat punnitusvenymäliuska- sekä paikoitusabsoluuttianturit. Koeajojen aikana parametroidaan paikoitukseen vaikuttavat tekijät, jotta liikkuvat koneenosat saadaan toimimaan mahdollisimman tarkasti, huomioiden käytettävissä oleva laskentateho ja I/O-kiertoaika. Näihin vaikuttavat logiikan ja I/O:n vasteajat tulee huomioida säätimien viritysparametrien ohella. Nykyaikaiset nopeat väylätekniikat auttavat paremman lopputuloksen aikaansaamisessa, mutta ohjelmoitavien logiikoiden suorituskyvyn puute aiheuttaa näyttämötekniikan liikkuvien koneenosien paikoitukseen haasteita.

Dokumentoinnin osalta näyttämöprojekteissa sovellussuunnittelija osallistuu nostimien kuormituskokeiden kirjaamiseen. Lisäksi pöytäkirjan ohjeistamana koeajoista kirjataan nostimien eri kuormilla suoritetut paikoituskokeet. Muita työtehtäviä ovat väylämittausten suorittaminen ja TLJ-liityntöjen tarkastaminen sekä dokumentointi. Tarkastuspöytäkirjadokumentti muodostaa tarkastuslistan kaikista käyttöönoton aikana suoritettavista tarkastuksista.

6.3.3 Ympäristö- ja vesihuolto

Vesihuoltoalalle suuntautuvissa pienissä automaatiotoimituksissa ei haastatteluaineiston perusteella juurikaan esiinny sovellussuunnittelun kelpoistuksen vaiheita. Kyseisellä toimialalla haastatteluaineiston perusteella oli havaittavissa rakenteellisesti kevyin sovellussuunnittelun prosessi. Tällä alalla myös asiakkaan vaatimukset laadunvarmistuksen osalta edustavat kevyitä menettelyjä. Näihin asioihin varmasti vaikuttavat johdannossa läpikäydyt asiat kohdeyrityksen kohdalla. Asiantuntijahaastettuluista ei myöskään tullut selkeästi ilmi tarvetta suunnittelutyövaiheelle. Vesihuoltoalalla kohdeyrityksen vahva asiantuntevuus ja kokemus pitkällä aikajänteellä on aiheuttanut sen, että suunnitteluprosessi on muotoutunut hyvin kevyeksi.

Määrittelyvaiheeseen voi liittyä toimintakuvauksen muodostaminen. Jos toimintakuvauksen toteuttaa kokenut suunnittelija, vastaa dokumentaatio ohjelmistokuvausta [1]. Tällaisessa tapauksessa toteutusvaiheen koodaaja voi kehittää automaatiosovelluksen toimintakuvauksen pohjalta. Tämän mahdollistaa yksiselitteinen semantiikkaa joka vastaa pseudokoodia. Etuna sovellussuunnittelijan muodostamasta toimintakuvauksesta nousee esille se, että koodia pystyy toteuttamaan kohdeproses-

[1] Liitteessä B esitellään automaation tukidokumentaatioksi soveltuva ohjelmistokuvauksen dokumenttipohja. Tämä on muodostettu hyvin jäsennellyn toimintakuvauksen pohjalta, johon on lisättä ohjelmistokuvaukselta vaadittuja asioita.

siin vähemmän perehtynyt sovellussuunnittelija. Muilta tehtäviltään sovellussuunnittelun prosessi vastaa muiden toimialojen yhteydessä esitettyä, kuitenkaan varsinaisia hyväksyntätestauksia ja kelpoistusjaksoja ei yleisesti suoriteta.

6.3.4 Energia- ja prosessiteollisuus

Tältä segmentiltä käsitellään ainoastaan energiateollisuutta, koska prosessiteollisuudessa edetään pitkälti aikaisemmin esitellyin menettelyin. Energiateollisuus edustaa raskaampaa sovellussuunnitteluprosessia. Tähän vaikuttaa osaltaan sovelluskohteiden turvallisuuskriittisyys ja tämän mukanaan tuoma TLJ-järjestelmän mukana olo. Asiantuntijahaastattelujen aikana kertynyt aineisto sisälsi kattavasti tehtäviä sovellussuunnittelun päävaiheiden osalta.

Määrittelyvaiheessa asiakkaan kanssa käydään keskustelua, jotta automaatiota käsittelevät periaatteet ja toimintakuvaus saadaan yksiselitteiseksi, aivan kuten muillakin toimialoilla. TLJ esiintyy suunnitteluvaiheen työnä, jossa kolmannen osapuolen toimintakuvauksesta toteutetaan standardin IEC 61131-3 mukaisella mallinnuskielellä turvaohjelman mallis. Tämän jälkeen turvaohjelman malli tarkastetaan riippumattomalla taholla. TLJ-sovelluksen toteutusosuus on suoraviivaista, koska lähtötietojen yksiselitteisyys ei jätä tulkinnan varaa. Tällöin kyseessä on käytännössä CAD-työkalulla piirretty graafinen esitys toteutettavan sovelluksen logiikasta, jonka sovellussuunnittelija implementoi ohjelmointikielelle.

Suunnitteluvaiheesta esille nousi I/O-luettelon tärkeys. Mikäli sitä ei ole määrittelyvaiheessa toteutettu, tulee se tehdä ennen suunnittelun aloittamista. Tämä asia pätee kaikelle automaation sovellussuunnittelulle. Käytännössä I/O-luettelo muodostetaan tarjolla olevasta dokumentaatiosta. Kyseessä on pohjimmiltaan sen asian selvittäminen, että mitä laitteita automaatioon kytketään. Sovelluksen arkkitehtuuri suunnitellaan, jotta aikaisemmin toteutettua työtä voitaisiin hyödyntää tulevissa projekteissa. Tämä on energiatekniikan projekteissa mahdollista, koska kohdeprosesseissa toistuvat samat laitekokonaisuudet. Muodostamalla suunniteltavasta sovelluksesta korkeamman abstraktiotason laiteriippumattoman esityksen, laajenee suunnitteludokumentaation hyödynnettävyys myös tulevissa projekteissa.

Toteutusvaiheessa käytetään generointityökaluja. Niiden avulla voidaan vähentää virheiden määrää, mutta haittapuolena haastatteluaineistossa mainitaan ohjelmointityön detaljien piiloon jääminen, jonka myötä aloittelevan suunnittelijan ymmärrys automaatiosovelluksen rakenteesta jää vähäisemmäksi. Toteutuksen päätteeksi suoritettavassa tehdastestauksessa tarkastetaan kaikki käyttöautomaation piirit. Energiateollisuudessa toimialalla tehdastestaukset ovat usein kattavampia kuin muilla toimialoilla. Tehdastestauksessa kenttälaitteet simuloidaan joko ohjelmasta tai ulkoisilla simulointisignaaleilla (milliampeeri- ja jänniteviesti).

Käyttöönotto lähtee liikkeelle mukaillen aikaisemmin kuvattua. Tämä vaihe al-

kaa järjestelmän pystyttämisellä ja vikatilojen selvittämisellä. Kenttälaitteiden I/O-testaus suoritetaan osana kylmätestausta, jonka jälkeen viritetään säädöt, testataan hälytykset ja turvapiirit. Tämän jälkeen siirrytään kuumatestaukseen ja mikäli erillistä hyväksyntätestausta ei ole, hyväksytetään automaation toiminta piirikohtaisesti tämän työvaiheen aikana. Hyväksyntätestaus, jota ei kaikilla toimialoilla vaadita, on energiateollisuudessa usein kuitenkin mukana. Hyväksyntätestauksen lisäksi suoritetaan kelpoistusjakso, jonka aikana automaatioon ei tehdä enää muutoksia. Pöytäkirjoja kirjataan tuttujen FAT ja I/O-testausten lisäksi myös säädöistä, johon on hyvä saada asiakkaan hyväksynnän merkiksi allekirjoitus.

6.4 Suunnitteluprosessista tunnistetut elinkaarimallien yhtäläisyydet

Kohdeyrityksen dokumentoitua sovellussuunnitteluprosessia tarkasteltiin sellaisista lähtökohdista, että osana tätä työtä muodostetaan kehitysehdotus kirjallisuudessa esitettyjen sovellussuunnittelun projekti- ja elinkaarimallien tarjoamilla konsepteilla. Dokumentoitu kohdeyrityksen sovellussuunnittelu istuu erittäin hyvin vesiputousmalliin, joka esiteltiin aikaisemmin luvussa 3.2. Alkuperäisen vesiputousmallin ongelmana tunnetaan vaatimusmuutosten jäykkyys johtuen vaatimusten jäädyttämisestä ennen toteutusvaiheeseen siirtymistä. Tähän lääkkeeksi on ohjelmistotuotannossa kehitetty iteratiivisiä ja inkrementaalisia ohjelmistoprojektimalleja.

Kohdeyrityksen projekteissa vaatimukset jäädytetään vesiputousmallin mukaisesti, jonka jälkeen voidaan aloittaa automaatiosuunnittelu. Automaatioprojektit toteutetaan suhteellisen nopeasti kauppojen syntymisestä, joten koko sovellus käytännössä tulee tehdä valmiiksi yhden syklin aikana, mikä osaltaan vastaa myös vesiputousmallin mukaista sovelluskehitystä.

Suunnitteluprosessin jäykkyys kesken projektin tapahtuviin muutoksiin onkin yksi merkittävimmistä haasteista automaatiosuunnittelussa johon ei yksinään voida vaikuttaa projektin elinkaarimallin valinnalla edellä kuvattujen projektien luonteenpiirteistä johtuen. Siksi muutoksiin varaudutaan jo sopimusta tehtäessä esimerkiksi laskuttamalla asiakasta erikseen lähtötieto- ja määrittelymateriaalin laajuuteen tulevista muutoksista.

Vaikka inkrementaaliset ja iteratiiviset ohjelmistotuotannon projektimallit vastaavat erityisen hyvin projektin aikana esiintyvien muutosten tuomiin haasteisiin, ei mallien mukaista toimintaa voida ilman tarkempaa suunnittelua (eikä tämän työn laajuudessa) lähteä toteuttamaan kohdeyrityksessä. Kohdeyritykselle tyypilliset automaatioprojektit ovat sovellussuunnittelun osalta resursoitu yhden suunnittelijan työpanokselle. Näin ketterien menetelmien mukaisen kehitystiimin tulisi kehittää samanaikaisesti useita eri projekteja, sillä muutoin kehitystiimi olisi ylimitoitettu.

Käyttöönotettuun järjestelmään voidaan kuitenkin sykleittäin tuoda päivityksiä, kuten laskennallisien arvojen laskentaa raportointia varten, vaihtoehtoisia ajotapoja tai vikakorjauksia. Kun järjestelmä on käyttöönotettu ja hyväksytty tuotantokäyttöön, voidaan tulevat muutokset käsitellä ylläpidon asioina ja toteuttaa päivityksinä. Automaatiosovelluksen ylläpitoon syklimäinen sovelluksen kehittäminen soveltuu teknisestä näkökulmasta hyvin. Siinä jokaiseen sykliin voidaan valita tietty määrä muutoksia, jotka voidaan toteutuksen jälkeen kelpoistaa yhdellä kelpoistusjaksolla. Tätä kuitenkin rajoittaa ylläpitosopimuksissa esiintyvät sovitut vasteajat, kuinka nopeasti vaadittuihin muutoksiin tulee puuttua.

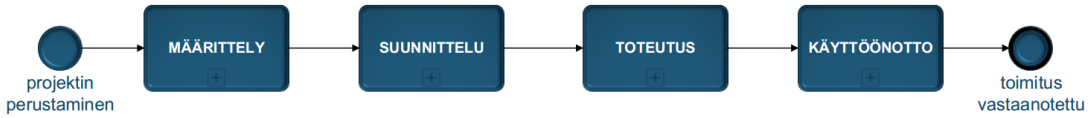
Vesiputousmallista tehdyt muunnelmat lähestyvät rakenteeltaan usein v-mallia. Tyypillisimpiä tällaisia ovat vesiputouksen jokaisen suunnitteluprosessin vaiheen varmistaminen jollain laadunvarmistuksen toimenpiteellä. Kohdeyrityksestä dokumentoitu sovellussuunnittelun prosessi vaikuttaakin muodostuneen luontaisesti v-mallin mukaiseksi sovelluksen testaamisen osalta, sillä se kulkee käsi kädessä toteutuksen edetessä. Myös v-mallissa esitetyt kelpoistamisen toimenpiteet ilmenevät haastatteluaineistosta. Kelpoistamisesta tehdastestaus on yleisesti projekteissa mukana oleva laadunvarmistuksen toimenpide, josta on sovittu jo sopimusta kirjoitettaessa.

Laadunvarmistustoimien olemassaoloon vaikuttavat omalta osaltaan asiakkaan vaatimukset kelpoistustoimien muodossa sekä hyvät käytänteet käyttöönoton onnistuneesta suorittamisesta. Poikkeavuuksia kohdeyrityksen toiminnassa esiintyy erityisesti määrittely- ja suunnitteluvaiheiden sisällössä sekä vaihetuotteiden todentamisessa. Vaikka toteutusvaiheessa tapahtuva testaaminen on v-mallin tyyppisesti osana sovelluksen kehitysprosessia, testaamisen objektiivisuuteen ei välttämättä kiinnitetä sen vaatimaa huomiota. Näihin asioihin on etsitty tässä ratkaisua ja seuraavassa kappaleessa puututaan havaittuihin asioihin kehittämisehdotuksen muodossa.

7. LAADUNVARMISTUSPROSESSIN KEHITTÄMINEN

Laadunvarmistusprosessin kehittämiseksi ensin dokumentoitiin kohdeyrityksen kokonaistoimitus -organisaation sovellussuunnitteluprosessit kaikista segmenteistä. Näistä koostettiin edellisessä luvussa esitetty yhtenäinen prosessimalli. Tässä luvussa dokumentoitua suunnitteluprosessia kehitetään eteenpäin laadunvarmistuksen näkökulmasta. Laadunvarmistamisen prosessin kehittämiseksi muodostetaan kehitysehdotus, jossa tuodaan esille nykyisen laadun varmistamiseen liittyviä toimenpiteitä. Ehdotus pohjautuu ohjelmistoteollisuuden kirjallisuudessa kuvattuihin asioihin sovellussuunnittelun sekä laadunvarmistuksen alueelta. Näiden avulla pyritään vaikuttamaan edellisessä luvussa huomioituihin haasteisiin sekä työlle asetettuihin tavoitteisiin. Näihin kuuluvat sovellussuunnittelun todentamisen objektiivisuus, suunnitteluvaiheen tukidokumentaation muodostaminen sekä laadun mittaaminen. Edellä esitetyt asiat nousivat tärkeimmiksi tekijöiksi kehityksen ensimmäisen askeleen ottamisessa sekä omalta osaltaan vaikuttavat myös laadunvarmistuksen lisäksi suunnittelutyön muihin osa-alueisiin.

Muodostettu esitys on tarkoitettu kohdeyrityksen hyödynnettäväksi ja siitä voidaan poimia projektikohtaisesti sopivimpia toimenpiteitä. Tällöin laadunvarmistuksen tarvetta tulee arvioida projektikohtaisesti ja valita projektin luonteeseen sopivimmat toiminnot tämän ehdotuksen rajaamalta alueelta. Laadunvarmistusprosessin etenemistä tarkastellaan tyypillisen skenaarion avulla prosessimallin soveltamisessa käytännön projektiin. Tässä esitetty OASP-malli kirjataan osana tätä työtä kohdeyrityksen laadunhallinnasta vastaavaan toimintajärjestelmään. Sen avulla suunnittelija voi perehtyä organisaation suunnitteluprosessiin, käyttää ohjeenaan dokumenttipohjia ja luetteloita vaadittavasta tukidokumentaatiosta. Lisäksi laadunvarmistuksen suorittaminen on ohjeistettu tätä tukemaan on muodostettu pohja sovelluksen tarkastamisesta. Tarkastuslistan dokumenttipohja on liitteenä A. Tähän kirjataan projektin tiedot, tarkastetut yleisimmät automaation sovellusprojekteissa esiintyvät osakokonaisuudet sekä tarkastajan valitsemat pistokoeluontoisesti tarkastetut toiminnallisuudet.



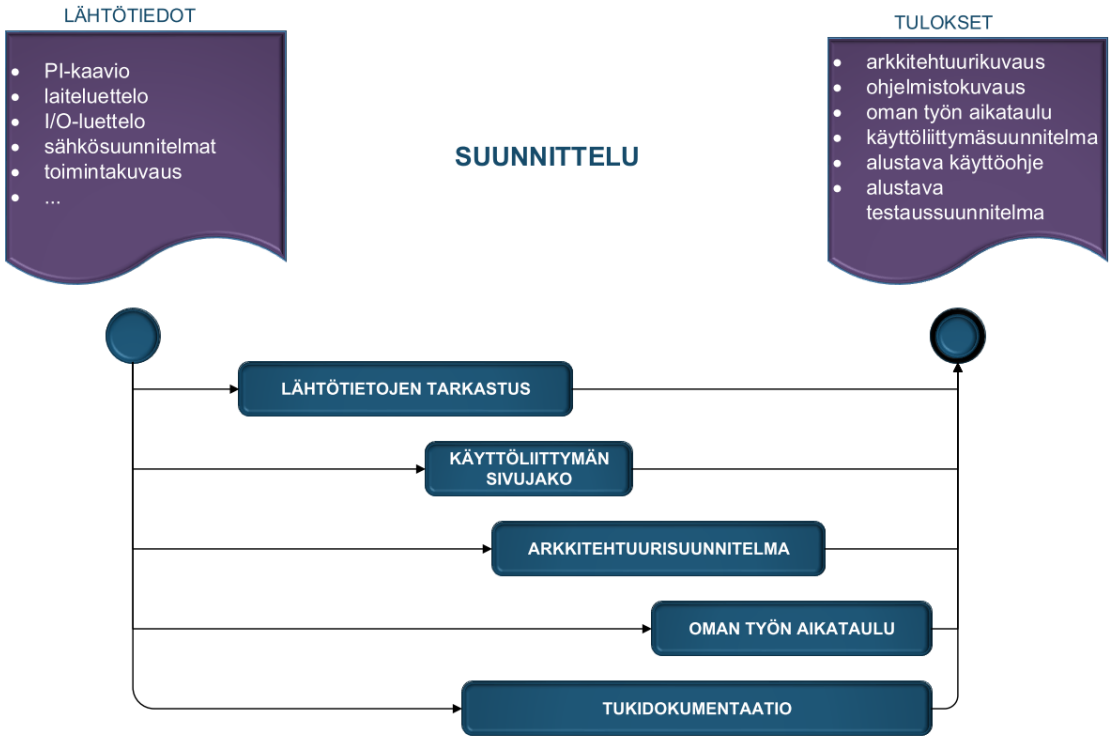
Kuva 7.1 *Sovellussuunnittelun päävaiheet.*

7.1 Ohjeellinen sovellussuunnittelun prosessimalli

Hyödynnettävään prosessimalliin ei tule luottaa sokeasti, vaan sen käyttökelpoisuus arvioidaan projektikohtaisesti. Arvioinnin tuloksena saadaan tarvittavat laadunvarmistustoimet ja näiden annostelu: tarkastellaanko kaikkia asioita yhdellä istunnolla vai projektin kuluessa pieninä annoksina, kuitenkin etukäteen muodostetun suunnitelman mukaisesti. Työprosessimalli voi tuntua itsestäänselvyydeltä, mutta näin ei kuitenkaan aina ole sillä mallista unohtuvat usein vaihekohtaiset arvioinnit, joita käytetään laadunvarmistuksen toimenpiteinä [42, s. 25]. Edellä kuvattu asia esiintyi siten, että asiantuntijahaastattelujen aikana jäi tulematta esiin sellaisia projektien aikaisia toimenpiteitä, jotka kokemusperäisesti ovat jossain määrin olemassa. Näitä toimenpiteitä ei ehkä yleisesti käsitetä työvaiheiksi tai laadunvarmistustoimiksi, eikä niiden suorittamista suunnitella eikä toteuteta systemaattisesti. Tässä luvussa kuvatulla ja kohdeorganisaation toimintajärjestelmään vietävällä ohjeellisella sovellussuunnittelun prosessimallilla pyritään vaikuttamaan vastaavanlaisiin asioihin.

Ohjeellinen OASP prosessimalli on muodostettu dokumentoidusta sovellussuunnittelun prosessista, jota on jatkokehitetty osana diplomityötä muodostetulla ehdotuksella laadusta varmistamisen osa-alueella. Kehitysehdotus tarjoaa selkeitä mutta sopivan kokoisia muutoksia dokumentoituun suunnitteluprosessiin. Tämä ohjeellisen prosessimalli pohjautuu vesiputousmallin ja v-mallin mukaisiin lähestymistapoihin (luku 3.2), joille luonteenomaisia piirteitä oli havaittavissa myös kohdeyrityksen toiminnasta dokumentoidusta suunnitteluprosessista luvussa 6.4. OASP-malli pohjautuu kirjallisuudessa esitettyjen sovellussuunnittelun vaihejakomallien päävaiheisiin. Näitä kuvaavat prosessit ovat esitetty kuvassa 7.1, joita käsitellään seuraavaksi.

Kehitysehdotuksessa korostetaan testaamisen objektiivisuudesta huolehtimista. Tämän lisäksi osana OASP-mallia määritellään tukidokumentaatio, jota suunnittelun aikana muodostetaan. Tukidokumentaation muodostaminen koetaan olevan suunnittelijoilla haasteellinen asia. Teoriaosuudessa näitä asioita käsiteltiin luvuissa 2.3.2 Suunnitteluvaihe sekä 5.4 Dokumentointi. Dokumentointia käsittelevän osuuden pääanti on taulukko sovellussuunnittelun tukidokumentaatiosta. Tämän taulukon esittämät dokumentit listataan toimintajärjestelmään, josta voidaan projekti-



Kuva 7.2 *Sovellussuunnittelun vaiheet ja sisäänmenodokumentaatio sekä saatavat tulokset.*

kohtaisesti poimia tarpeellisimmat.

Tukidokumentaation tarjoamasta annista korostetaan ohjelmistokuvausta joka tässä asetetaan muodostamaan portti, jonka kautta kaikki sovellukseen implementoitavat asiat tuodaan. Tämän myötä jokaisen uuden sovellusprojektin aikana tuotetaan ajan tasalla pysyvä sovellusta käsittelevä tukidokumentti, jonka tuomia hyötyjä käsitellään tarkemmin omassa kappaleessaan. Ohjelmistokuvaus sisältyy suunnitteluvaiheen tehtäväksi ja se käsitetään toteutettavaksi osana kuvassa 7.2 mainittavaa tukidokumentaation muodostamistehtävää. Vaikka arkkitehtuurisuunnitelma voidaan ajatella osaksi ohjelmistokuvausta ja asettuu näin tukidokumentaation muodostamistehtävän alle, halutaan mallissa korostaa tämän tärkeyttä erittelemällä se omaksi tehtäväkseen.

Suunnitteluvaihetta esittävässä työprosessissa on nähtävillä suunnitteluvaiheen sisäänmenotiedot sekä suunnitteluvaiheen tehtävien tuloksena saatavat mahdolliset tulokset. Muut suunnitteluvaiheen tehtävistä kuvattiin aikaisemmin kohdeyrityksen sovellussuunnitteluprosessin dokumentointia käsittelevässä luvussa 6.2.

Ohjeellinen toteutusvaihe mukailee kirjallisuuden esitystä (luku 2.3.3), jota on muokattu soveltumaan paremmin kohdeyrityksen valtavirran automaatiosovellusprojekteihin. Toteutusvaiheen sisältö kuvassa 7.3 vastaa hyvin pitkälti asiantuntijahaastattelujen mukaista aineistoa. Kaaviokuvassa esitetään kuinka automaatioso-

velluksen käyttöliittymän asetusarvosivut on hyvä toteuttaa vasta siinä vaiheessa, kun logiikkaohjelma on valmis. Näin toimitaan, koska tässä vaiheessa tiedetään kaikki tarvittavat asetusarvot toiminnallisuuksien parametroimiseksi. Ohjeellisessa mallissa ei ehdoteta suunnittelemaan moduuleja yksityiskohtaisesti etukäteen jokaisen moduulin osalta, koska automaatiosovelluksen suoraviivainen luonne ei tätä vaadi. Ainoastaan haastavimmat tulisi suunnitella ennen toteutusta. Tällä pyritään pitämään sovellussuunnittelu mielekkäänä minimoimalla byrokratiaa.

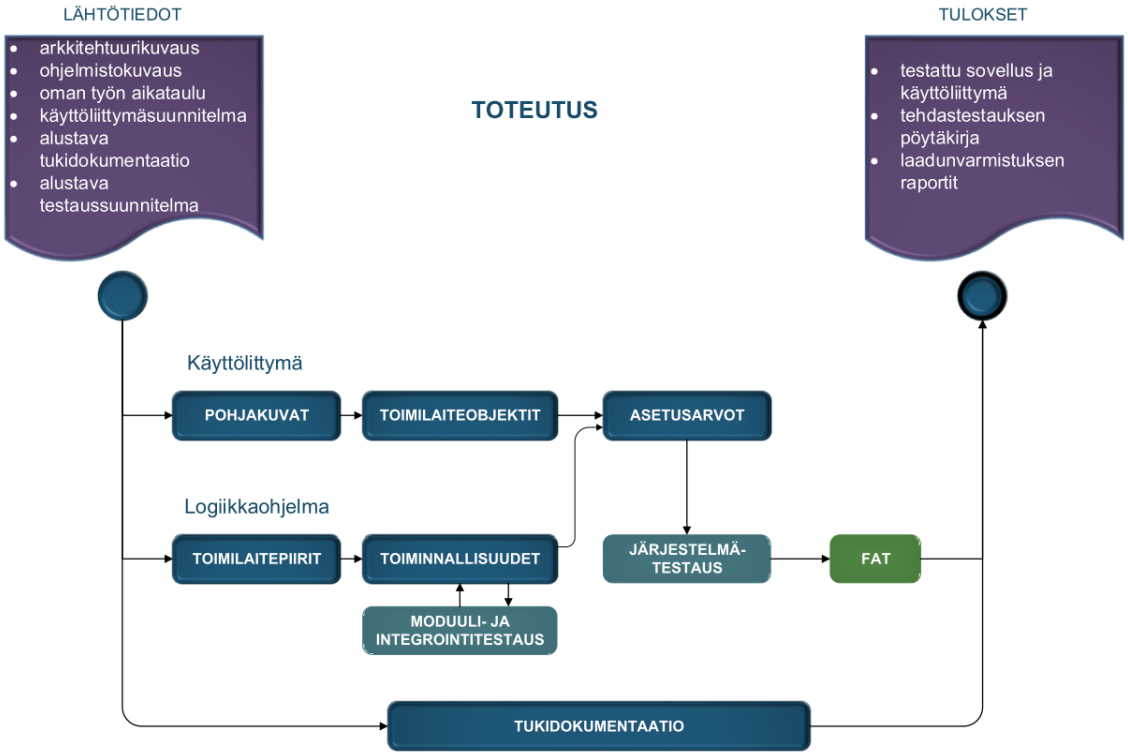
Merkittävänä lisäyksenä kaaviokuvaan 7.3 lisätään järjestelmätestaus. Kyseessä on kirjallisuudessa esitetty vaihejakomalleissa esiintyvä toimenpide, joka kohdeyrityksen projekteissa on välillä unohtunut. Järjestelmätestauksella varmistetaan kokonaisuuden olevan esittelykelpoinen asiakkaalle/loppuasiakkaalle esiteltäväksi. Tehdastestauksen aikana esiintyvät pienet virheet eivät ole merkittäviä ja usein korjattavissa ennen käyttöönottoon siirtymistä. Ei kuitenkaan anna hyvää kuvaa yrityksestä eikä herätä luottamusta, mikäli kriittiset osat prosessinohjausta on ymmärretty väärin tai toteutus on keskeneräinen. Näihin asioihin järjestelmätestaus puuttuu tehokkaasti.

Edellisen suunnitteluvaiheen tukidokumentaatio on työtehtävänä myös toteutusvaiheen kaaviossa. Käytännössä tämä tarkoittaa tukidokumentaation muodostamisen jatkamista. Dokumentoinnin tehtävinä voivat olla ohjelmistokuvauksen päivittäminen toteutuksen edetessä tai käyttöohjeen muodostaminen käyttöliittymän kuvakaappauksilla varustettuna. Seuraavassa luvussa otetaan käsittelyyn ohjeellisen mallin mukanaan tuomia laadunvarmistustoimia.

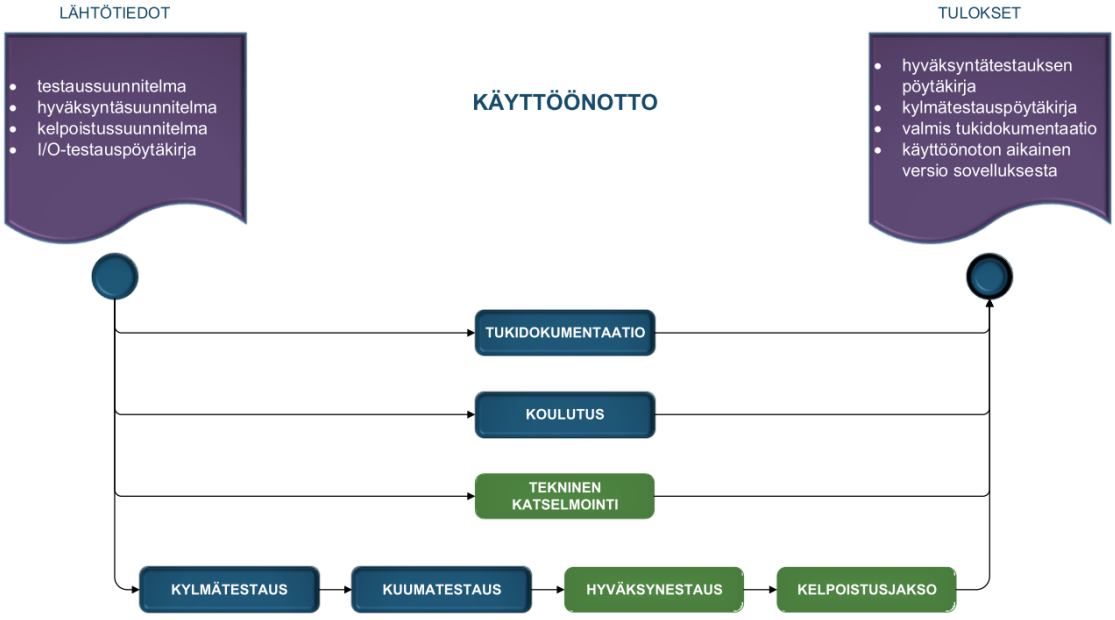
7.1.1 Laadunvarmistusprosessi osana ohjeellista prosessimallia

Hyödynnettäessä muodostettua prosessimallia käytännössä mallissa esitetyistä laadunvarmistuksen tehtävistä sekä tuotettavasta dokumentaatiosta valitaan kyseisen projektin luonteeseen sopivat projektin alkuvaiheessa. Laadunvarmistuksen kohdentaminen ja sopivat tukidokumentit täytyy arvioida tapauskohtaisesti ja valintoihin vaikuttavat toimialalla esiintyvät vaatimukset sekä kehitysprosessiin osallistuvan suunnittelijan perehtyneisyys. Asiakas voi esimerkiksi tarvita todistusaineistoa saavutetusta laadusta, jota käytetään automaatiojärjestelmän sekä koko kohdeprosessin kelpoistamisessa. Todistusaineistoa saadaan aikaiseksi seuraamalla kappaleessa 7.1.3 *Laadunvarmistusmekanismien avulla tuotettavat laadun todisteet* esitettyjä toimenpiteitä. Kehitysehdotus sisältää valinnanvapauden suoritettavan byrokratian suhteen ja suunnitteluprosessin lisäksi myös laadunvarmistusprosessi voidaan pitää mahdollisimman kevyenä. Käytäntöön soveltamisesta on esitetty luvussa 7.2.

Vähemmän perehtyneen sovellussuunnittelijan tapauksessa laadunvarmistuksen



Kuva 7.3 *Kehitysehdotuksen mukainen esitys toteutusvaiheesta, lisäksi esitetty vaiheen lähtötiedot sekä tulokset.*



Kuva 7.4 *Käyttöönoton jälkeen laadunvarmistuksellisena toimenpiteenä suoritetaan tekninen katselmointi.*

tehtävät valitaan siten, että saavutetaan tavoiteltava laadun taso sekä täytetään koulutukselliset näkökohdat. Tällöin tehtäviksi valitaan esimerkiksi suunnittelun ja toteutuksen verifioiminen vertaisarvioinnilla. Projektin toimialaan ja/tai valittuun toteutusteknologiaan paremmin perehtynyt kokenut suunnittelija suorittaa verifioimisen läpikäyntien avulla luvussa 5.1.2) kuvatusti. Todennettavia asioita tässä mahdollisimman kevyessä laadunvarmistusprosessi ovat:

- Suunnittelijan muodostaman ohjelmistokuvauksen tarkastaminen lähtötietoja vastaan
- Toteutetun koodin läpikäynti
- Tukidokumentaation tarkastaminen
- Järjestelmätestaus ohjelmistokuvausta vastaan

Suunnitteluvaiheen jälkeen ennen toteutukseen siirtymistä tarkastetaan sovelluksen arkkitehtuurisuunnittelu osana ohjelmistokuvauksen tarkastusta. Ohjelmistokuvauksen tulee perustua täysin lähtötietoihin ja arkkitehtuurissa otetaan huomioon mahdolliset hajautukset ja moduulien uudelleenkäyttö. Näihin asioihin löytyy kokeneilla suunnittelijoilla näkemystä. Seuraavan vaiheen aikana voidaan läpikäynnin aikana tarkastaa toteutus. Tällöin tarkastellaan että suunnittelija on implementoinut toimilaitepiirit tarkoituksenmukaisesti, sovelluksen toiminnallisuuksien läpikäynnissä tarkastellaan suunnittelijan koodauskäytänteet ja -kommentointi. Läpikäyntiä käsittelevässä luvussa mainittiin myös, että osana läpikäyntiä suunnittelija selittää miten luulee koodinsa toimivan. Näin arvioijalle jää parempi käsitys työn oikeellisuudesta. Toiminnallisuuksien tarkka detaljikohtainen tarkastaminen ei useinkaan ole mahdollista koodin suuresta määrästä johtuen. Tilaisuus tarjoaa kuitenkin hyvän mahdollisuuden opastaa vaihtoehtoisia koodauskäytänteitä sekä oikeata testausmenettelyitä moduuli- ja integrointitestaukseen.

Tavallisissa automaatioprojekteissa sovelluksen toimivuuden testaa itse suunnittelija. Automaatiosovelluksen luonteesta johtuen testaamisessa keskitytään todentamaan koodin toimivuus eikä siitä voida etsiä kaikkia mahdollisia virhetilanteita kuten ohjelmistotuotannon testaustoimenpiteillä pyritään. Tämä johtuu automaatiototeutusten reaaliaikaisuudesta jossa ulkoisten syötteiden mahdollisia eri ajoituksia on lukematon määrä, joten näiden todentaminen simulointiympäristössä muodostuisi todella laajaksi toimenpiteeksi, eikä välttämättä olisi edes mahdollista. Testauksen käytänteet tämän osalta ovat vaihdelleet projektikohtaisesti.

Asiakkaan kanssa pidettävässä tehdastestauksessa (luku 6.2.1) kelpoistetaan automaatiototeutus, jotta järjestelmä hyväksytään toimitettavaksi. Luvussa 5.1.4 kuvattu järjestelmätestaus esitellään osana laadunvarmistuksen kehitystä. Tässä kuvatussa ohjeellisessa mallissa se tapahtuu ennen tehdastestauksen aloittamista. Tähän

testauksen vaiheeseen kytketään laadunvarmistuksen toimenpide, jonka avulla vastataan testauksen objektiivisuusvaatimukseen. Järjestelmätestauksessa toteutuksesta ulkoinen henkilö tarkastaa järjestelmän yhtenevyyden vaatimuksiin ja lähtötietoihin osana järjestelmätestausta ennen tehdastestaukseen siirtymistä. Tarkastustoimenpide tulee ottaa mukaan projektisuunnitelmaan sekä resursoida projektin luonteen mukaan. Tarkastajana toimii vaadittavan perehtyneisyyden omaava työntekijä, jolla on edellytykset toimia projektin pääsuunnittelijana/tarkastajana.

Tähän laadunvarmistustoimenpiteeseen lisätään kirjanpito ja raportointi virheistä, jollaista ei ole aikaisemmin totuttu tekemään. Raporttien avulla voidaan seurata tietyntyyppisten projektien virhemääriä ja virhetyyppejä. Ennen tarkastusten käyttöönottoa tuleekin määritellä yleisimmät automaatiosovelluksissa esiintyvä virheet, joihin havaitut virheet luokitellaan. Näistä tiedoista pystytään seuraamaan sovellussuunnittelun kehitystä virhemäärien osalta ja tehdä korjaavia toimenpiteitä suunnitteluprosessia jatkossa kehitettäessä. Tähän käytetään kaikkia projektista suoritettuja mittausarvoja. Näitä ovat tällä hetkellä suunniteltujen aikataulujen toteutuminen, arvioidun ajankäytön toteutuminen sekä asiakastyytyväisyyskysely, ja tulevaisuudessa projektikohtaiset virhetilastot.

Projektinhallinnan toimenpide *tekninen katselmointi*, joka käsiteltiin kappaleessa 5.1.3, otetaan mukaan projektinhallintaan käyttöönoton jälkeen tapahtuvaksi (kuva 7.4). Ohjelmistoteollisuudessa toteutus katselmoidaan ennen käyttöönottoon siirtymistä, mutta automaatiosovellukset muuttuvat niin paljon käyttöönoton aikana, että tekninen katselmointi on syytä sijoittaa vasta tämän jälkeen. Se voidaan myös pitää ennen sekä jälkeen käyttöönoton. Projektin loppupuolella pidettävässä katselmoinnissa varmistetaan, että kaikki projektiin liittyvä sovellussuunnittelu on toteutettu ja dokumentaatio on ajan tasalla eikä projektista jää "häntiä", kun riennetään kiireellä seuraavaan projektin pariin.

Käytännössä koulutus suoritetaan käyttöönoton yhteydessä sovellussuunnittelijan ollessa jo valmiiksi asiakkaan/loppuasiakkaan luona. Koulutuksessa tukimateriaalina ja koulutussuunnitelmana toimii automaatiosovelluksen käyttöohje. Koulutuksen aikana tulee keskittyä case -tyyppiseen koulutukseen, jossa koulutettavat opetetaan suorittamaan tiettyjä operointeja käyttöliittymän avulla. Lisäksi koulutettavat tulee haastaa suoriutumaan opetetuista asioista omatoimisesti, koska pääasiassa tulee olla prosessin operointi eikä käyttöliittymän yksityiskohdat. Käyttöönottoa käsiteltiin jo dokumentoidun sovellussuunnittelun yhteydessä, eikä tässä ole aiheeseen mitään lisättävää. Käyttöönotto on muodostunut eri toimialasegmenttien kohdeprosessien ja käytänteiden mukaiseksi, eikä erityisiä kehitystarpeita tähän vaiheeseen teknisen katselmoinnin lisäksi ole.

OASP-malli esitettiin tässä prosessikaaviona, kuvat 7.1, 7.2, 7.3 sekä 7.4. Prosessikaaviossa esitetään tehtävien suorittaminen sekventiaalisena etenemisenä. To-

dellisuudessa tilanteet voivat poiketa tässä esitetystä suoritusjärjestyksestä, mutta johdonmukainen eteneminen selkeyttää kuvattua prosessia, sekä esittää toivottavan työnkulun järjestyksen. Erityisesti kaaviot toimivat perehdyttävinä esityksinä uusille suunnittelijoille.

7.1.2 Ohjelmistokuvauksen merkitys

Ohjelmistokuvaus kirjoitetaan yleensä ylläpitäjää ajatellen siitä näkökulmasta, että toteutettua ohjelmaa halutaan myöhemmin muuttaa. Tästä dokumentista tulisi siis löytyä kaikki tarvittava tieto sovelluksesta. Automaation sovellusprojektien luonteesta johtuen projektinaikaisten muutosten todettiin olevan kovin yleisiä luvun 5.2.2 alussa. Tästä johtuen muutosten dokumentointi tulee ottaa mukaan sovellussuunnittelun työprosessiin. Ennen toteutusta muutokset kirjataan ylös ohjelmistokuvaus -dokumenttiin, jonka lokiin lisätään tieto muutoksista. Kuvassa 7.5 esitetään sovelluskehityksen aikaisista vaiheista esiintyneiden muutostarpeiden kierrättäminen suunnitteluvaiheen kautta, jossa hyväksytyt muutokset suunnitellaan ja kirjataan ohjelmistokuvaukseen toteutusta varten.

Ohjelmistokuvaus on tarkoitettu ainoastaan sovellussuunnittelijoiden käytettäväksi, joten se voidaan muodostaa käyttämällä vapaasti teknistä kieltä eikä maallikoiden tarvitse ymmärtää dokumentin antia. Sovelluksen rakennetta kuvataan tässä dokumentissa toteutustasolla. Kyseessä on toimintakuvauksen ja ajotapakeskustelujen perusteella yksiselitteiseksi kuvaukseksi muodostettua luonnollista kieltä sekä pseudokoodia, joka voidaan muuttaa ongelmitta ohjelmakoodiksi [31]. Ohjelmistokuvauksen dokumenttipohja viedään yrityksen toimintajärjestelmään. Hyödyntämällä tätä dokumenttipohjaa, tulevaisuudessa tehtävät ohjelmistokuvaukset tulevat olemaan yhtenäisiä ja selkeitä. Dokumentti ohjeistaa kirjaamaan oleellisimmat asiat, jolloin ohjelmistokuvaus pysyy selkeänä ja lyhyenä. Tämä tukidokumenttipohja on nähtävissä liitteenä B.

Ohjelmistokuvauksen muodostamisessa suunnittelija kirjoittaa suoraviivaisella lähestymistavalla lähtötietojen perusteella sovelluksen suunnitelman. Tämän aikana suunnittelija perehtyy automatisoitavaan kohdeprosessiin, jolloin ymmärrys kokonaisuuden toiminnasta aina yksityiskohtia myöden selkeytyy. Samalla nousee esiin jatkoselvitystä kaipaavia asioita ja näihin päästään etsimään vastauksia heti projektin alkuvaiheessa. Tämä vähentää loppuvaiheessa esiin tulevia yllättäviä ongelmia, jotka voivat aiheuttaa aikatauluhaasteita. Kun suunnittelija on ajatellut toteutuksen jo aikaisemmin, nousee hänellä mieleen vaihtoehtoisia ja mahdollisesti selkeämpiä toteutusvaihtoehtoja projektin edetessä toteutusvaiheeseen. Tällöin toteutuksesta saadaan laadukkaampi esimerkiksi ylläpidettävyyden tai testattavuuden osalta.

Kappaleessa 4.7 mainittiin projektien henkilöitymisen luovan ongelmia erityistilanteissa, joissa suunnittelija saa tehtäväkseen jatkaa toiselta kesken jäänyttä työtä.

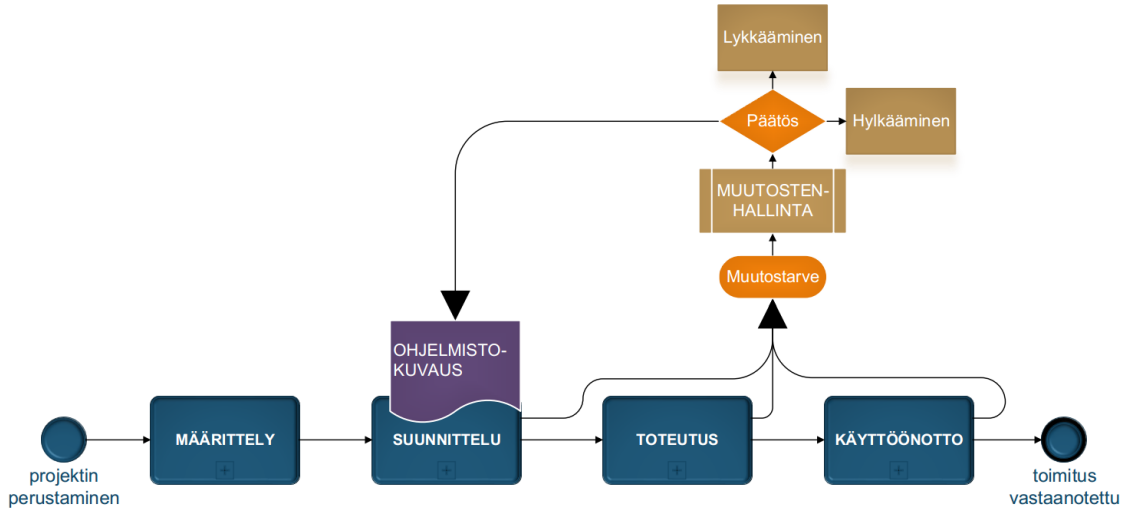
Lisäongelmia tällaiseen tilanteeseen tuo alkuperäisen suunnittelijan saavuttamattomuus, esimerkiksi sairastumisen tai työpaikan vaihdon seurauksena. Kun projektin suunnitteluvaiheessa on luotu ohjelmistokuvaus, on projektiin tulleen seuraavan suunnittelijan mahdollista muodostaa käsitys ajatuksesta jolla sovellus on suunniteltu toteutettavaksi. Näin keskeneräisestä työstä saa helpommin otteen mitä siitä vielä uupuu ja on toteuttamatta. Tällöin suunnittelija pystyy myös jatkamaan sovelluksen kehitystä ilman, että tähän asti tehtyä työtä pitäisi heittää pois. Tällöin suunnitteluun käytetty aika maksaa itsensä takaisin moninkertaisesti. Toinen merkittävä resursseihin liittyvä lisähyöty ohjelmistokuvauksesta tulee esille laajassa tai kiireisessä projektissa, jonka toteuttamisvaiheeseen liittyy auttamaan projektia aikaisemmin tuntematon suunnittelija. Tällöin projektin sovellussuunnittelu voidaan viedä eteenpäin kahden työntekijän voimin ilman, että projektiin liittyneen opastamiseen kuluu turhan paljoa aikaa. Suunnittelijoiden tehtävänä on suoraviivaisesti toteuttaa määritelty kokonaisuus ohjelmistokuvausta noudattamalla.

Ohjelmistokuvauksessa voi olla alla kuvattuja asioita [31]:

- johdanto
- järjestelmän yleinen rakenne
- sovelluksen arkkitehtuuri
- toimintakuvauksen otsikoinnin mukainen kuvaus sovelluksen toiminnallisuuksista
- kuvauksen mukaiset hyödynnettävät osakokonaisuudet (automaatiosovelluksessa näitä ovat sovelluksen toimilaitepiirit, asetusarvot ja hälytykset sekä lukitukset)
- tietokantojen ja taulujen väliset yhteydet

Sovellussuunnittelijan ei ole tarkoitus rientää koodaamaan ennen kuin tämä dokumentti on toteutettu ja lähtötiedoissa esiintyneet epäselvyydet ovat selvitetty. Erityistä huomiota vaatii kuitenkin se realiteetti, että aikataulu ei projekteissa useinkaan kestä kaikkien vaadittavien tietojen odottamista asiakkaalta. Tämä voi johtua vielä kesken olevasta asiakkaan prosessisuunnittelusta tai laitehankinnoista. Kaikkea tarvittavaa tietoa ei yksinkertaisesti ole vielä saatavilla. Tällöin sovelluksen tekeminen usein pitää aloittaa puutteellisin tiedoin joita täydennetään projektin kuluessa, viimeistään käyttöönotossa selviää tuntemattomat asiat. Asiakkaan kanssa keskustelemalla voidaan kuitenkin löytää seikkoja jotka todennäköisimmin eivät muuta ja joiden myötä voi hyvillä mielin siirtyä toteutusvaiheeseen.

Ohjeelliseen prosessimalliin sijoitettiin tässä käsiteltävä ohjelmistokuvaus portiksi sovellukseen tehtäville muutoksille. Kuvan 7.5 mukaan muutostarve voi esiintyä



Kuva 7.5 *Ohjelmistokuvaus toimii porttina, jonka kautta sovellukseen tehdään toteutuksia ja projektin aikaisia muutoksia.*

suunnittelu- ja toteutusvaiheen aikana sekä käyttöönotossa. Nämä kulkevat muutostenhallinnan kautta, josta hyväksytyt muutokset kierrätetään ohjelmistokuvauksen kautta toteutettavaksi. Kohdeyrityksen ylläpito- ja elinkaaripalveluiden asiantuntijoiden haastattelussa [35],[52] kävi ilmi, että muutostenhallinta perustuu projektipäällikön omien käytänteiden mukaiseen toimintaan. Tähän liittyvät muutoksista pidettävä henkilökohtainen historialoki sekä laskutuksen kannalta tallenteita tehdystä työstä. Tämän työn luvussa 5.5 käsitellään teoreettisemmasta näkökulmasta muutostenhallinnan prosessia, jossa siihen sijoittuvat myös muutoksen analysointi sekä kelpoistaminen.

7.1.3 Laadunvarmistuksen mekanismeilla tuotettavia laadun todisteita

Aikaisemmin esitetyssä ohjeellisessa prosessimallissa olevilla laadunvarmistuksen toimenpiteillä voidaan tuottaa todisteita saavutetusta laadusta. Näiden avulla voidaan esittää asiakkaalle dokumentaatio tuotteesta ja sen oikeellisuudesta. Aikaisemmin on mainittu, että todisteita käytetään järjestelmän kelpoistamiseen. Laadunvarmistukseen liittyvistä tehtävistä ja tuotetuista dokumenteista on koottu yhteenveto taulukkoon 7.6. Osa dokumenteista mainittiin jo sovellusprojektin dokumentaatiota käsittelevässä alaluvussa 5.4.

Määrittelyjen todentamisessa on kyse vaatimusten käytännöllisyyden ja tarpeellisuuden selvittämisessä sekä asiakkaan kanssa yhteisymmärryksen muodostamisesta. Määrittelyvaiheen merkittävimpänä tehtävänä on selvittää lähtötietodokumenteista puutteet ja moniselitteisyydet. Jos määrittelyvaiheeseen liittyy suunnittelutyönä so-

Kuva 7.6 *Automaatioprojektin aikana muodostettavia laadun todisteita.*

laadunvarmistuksen tehtävä	**laadun todisteet**
suunnittelun aikana muodostetun tukidokumentaation tarkastaminen	muodollisen katselmoinnin tai tarkastuksen pöytäkirja
käyttöliittymän graafisen esityksen hyväksyttäminen asiakkaalla	kirjallinen asiakaspalaute käyttöliittymän osalta
sovelluksen testaaminen lähtötietoja vastaan	testausraportit ja järjestelmätestauksen virheraportti
sähköautomaatiokeskuksen I/O-testaus	I/O-testauspöytäkirja
tehdastestaus	tehdastestauspöytäkirja
kylmä- ja kuumatestaus	piirikohtainen I/O-testauspöytäkirja
hyväksyntätestaus	hyväksyntätestauspöytäkirja
käyttöönoton jälkeinen tekninen katselmointi	katselmoinnin pöytäkirja

velluksen lähtötietojen kehittämistä, tuotetut dokumentit kelpoistetaan katselmoimalla ne asiakkaan kanssa.

Suunnitteluvaiheen aikana toteutettu sovelluksen suunnittelu- ja tukidokumentaatio todennetaan vertaamalla sitä lähtötietojen dokumentaatioon. Objektiivisen tarkastuksen avulla todisteita saadaan luotua tarkastusraporttien muodossa. Ohjelmistokuvaus ja arkkitehtuurisuunnittelu voidaan verifioida, jotta kaikki vaatimusmäärittelyspesifikaation mukaiset toiminnalliset vaatimukset ovat varmasti huomioitu tuotetussa dokumentaatiossa. Tästä on syytä varmistua, koska sovellus toteutetaan näiden perusteella. Toteutunut jäljitettävyys voidaan ottaa mukaan laadunvarmistuksen piiriin. Jäljitettävyys todennetaan tarkastamalla suunnitteluvaiheen aikana muodostettu jäljitettävyysmatriisi arkkitehtuurisuunnittelun ja vaatimusten välillä.

Toteutusvaiheen aikana laadun todisteita voidaan muodostaa tarkastamalla koodia muodollisien menettelyjen avulla, aivan kuten suunnitteluvaiheessa tarkastetaan dokumentaatiota. Testauksesta saadaan todisteet testausraporttien muodossa. Tämä kuitenkin vaatii testauksen suunnittelun. Tähän käytetään etukäteen muodostettuja testitapauksia, joissa toimintakuvauksen mukaisesta toiminnasta varmistutaan. Hyväksytysti raportoiduilla testitapauksilla voidaan muodostaa laatutodisteita moduuli- ja integrointitestausvaiheista, sekä järjestelmätestauksesta.

Ennen toimitusta sähköautomaatiokeskukset testataan riviliitinrajapinnasta. Asiakas voi osallistu tehdastestaukseen. On täysin projektista riippuvaa kuinka kattava tehdastestauksesta muodostuu. Pääasiassa simuloidaan mittauksia tai toimilaitteita keskuksen riviliitinrajapinnasta ja esitellään käyttöliittymästä prosessin operointimahdollisuuksia. Tehdaskoestuksessa on kyse asiakkaan suorittamasta kelpoistamistoimesta, jossa asiakas tutustuu toimitettavaan järjestelmään ensimmäisen kerran.

Ennen koeajojen aloittamista varmistutaan siitä että kenttälaitteet ovat kytketty ja parametroitu oikein, sekä mahdollinen automaatioväylä saatetaan toimintaan. Kaikki kenttälaitteet (instrumentit, moottorit ja venttiilit) testataan ja ne laitetaan toimintakuntoon yksitellen. Koeajojen aikana kirjataan esimerkiksi ajan tasalla olevaan toimintakuvaukseen testatut piirit sekä mahdolliset huomiot. Jos erillistä hyväksyntätestausta ei järjestetä, otetaan testauspöytäkirjaan asiakkaan edustajan allekirjoitus järjestelmän hyväksynnän merkiksi automaation osalta. Varsinaisessa hyväksyntätestauksessa voidaan toimia samoin tai käytetään erillistä hyväksyntäpöytäkirjaa. Käyttöönoton jälkeen katselmoidaan projektin tulokset ja tuotokset tekniseksi katselmoinniksi nimetyssä tilaisuudessa.

7.2 Ohjeellisen prosessimallin soveltaminen eräässä automaatioprojektissa

Tämän työn aiheena muodostettua sovellussuunnittelun dokumentoitua prosessimallia sovelletaan erääseen käynnissä olevaan projektiin. Kyseessä on ohjeellisen prosessimallin hyödyntäminen, eikä niinkään raportoitu soveltuminen jo suoritettuun kokeiluun. Myöskään suunnitelmaa ei ole tarkoitus toteuttaa tämän projektin osalta. Tässä luvussa esitellään teknisestä näkökulmasta yleisellä tasolla automaatiotoimituksen rakenne, sovitetaan sovellussuunnittelun prosessimallia sekä lopuksi esitetään asiantuntijan arvio soveltuvuudesta. Arvion avulla pyritään saamaan palautetta kehitysehdotuksen mahdollisista vahvuuksista ja sen mukanaan tuomiin haasteisiin, joita voidaan hyödyntää kehitysehdotusta käyttöönotettaessa.

Projektoinnissa muutostenhallintaan pyritään vaikuttamaan sopimalla ennen käyttöönoton alkamista vertailutilaisuus asiakkaan kanssa, jossa tarkastellaan tilauksen ja toteutuneen toimituksen eroavaisuudet. Lisäksi tilauksesta poikkeavat työt sovitaan kirjallisella muutostenkäsittelyllä sekä näiden toteuttaminen laskutetaan erikseen tuntiveloituksella. Asiakkaalta lähtötietoina saadut määrittelydokumentit sisältävät PI-kaaviot, laiteluettelot sekä laitteiden sijoituskuvat. Asiakas velvoitetaan ylläpitämään edellä mainittuja, jos/kun muutoksia esiintyy. Käyttöönottovaiheeseen liittyvä kenttälaitteiden parametrointi kuuluu projektissa asiakkaan vastuualueeseen.

Automaatioprojekti edustaa kohdeyrityksen kokonaistoimitusyksikölle tyypillistä projektia laajuutensa suhteen. Projekti esitellään tässä yleiseltä kannalta menemättä yksityiskohtiin. Automaatiototeutus koostuu ohjelmoitavan logiikan ja komponenttipohjaisen valvomojärjestelmän sekä kosketuspaneelin laitteistosta. Toimilaitteiden välinen kommunikointi hoidetaan automaatioväylän avulla sekä hajautettujen I/O-yksiköiden avulla. Lisäksi automaatioväylän välityksellä liitetään toimitettava järjestelmä ulkopuolisiin järjestelmiin. Projekti koostuu yhteensä noin 400:sta kap-

paleesta kenttäliityntöjä instrumenttien, venttiilien sekä moottorikäyttöjen osalta.

7.2.1 Laadunvarmistussuunnitelma

Automaatiosovelluksen toteuttamiseksi ja käyttöönottamiseksi tässä esitetään suunnitelma jossa huomioidaan sopimuksen myötä aikaisemmin määritellyt sekä ohjeellisen mallin mukaiset mahdollisimman kevyet laadunvarmistustoimet (tukidokumenttien tarkastus, koodin läpikäynti, järjestelmätestaus). Laadunvarmistussuunnitelma on muodostettu kohdeyritykselle tyypilliseen automaatioprojektiin, jossa sovelluksen kehitys toteutetaan saatujen lähtötietojen perusteella ja näitä täydennetään ajotapakeskustelujen avulla.

Arvioitavana olevassa projektissa ei esiinny määrittelyvaiheelle tyypillisiä asiakasvaatimusten kartoittamisia tai vaatimusmäärittelyn muodostamista, jotka vaatisivat katselmointia tai tarkastamista. Projektin osalta määrittelyvaihe toteutuu kolmannen osapuolen (konsultin) toimesta. Sovellussuunnittelijan tehtäväksi jää ainoastaan lähtötietoina saatujen dokumenttien tarkastamista ja epäselvyyksien selvittäminen asiakkaan kanssa yhteistyössä. Arvioitavana olevassa projektissa se tarkoittaa PI-kaavion, toimintakuvauksen sekä laiteluettelon keskenään ristiin vertaamista. Ennen suunnittelun varsinaista aloittamista saavutetaan ymmärrys automatisoitavan prosessinhallinnan ajatuksesta.

Seuraavan vaiheen aikana käydään ajotapakeskustelua asiakkaan kanssa sekä muodostetaan ohjelmistokuvaus, jonka perusteella sovellusta lähdetään toteuttamaan. Ohjelmistokuvaus antaa suuntaa toteutettavalle arkkitehtuurisuunnittelulle, jossa toteutuksen sisältö jaetaan loogisiin kokonaisuuksiin. Vaiheen aikana ei toteuta erillistä jäljitettävyysmatriisia yhdistämään toimintakuvauksen mukaiset asiat suunnittelu- tai lähtötietodokumentaatioon. Näin voidaan toimia aina, kun projekti on kokoluokaltaan pieni sekä toiminnallisuuksien keskinäinen vuorovaikutus on vähäistä. Jäljitettävyyteen vaikutetaan kuitenkin käyttämällä ohjelmistokuvauksen otsikkoina ja sisällön jakona toimintakuvauksen vastaavaa. Yhtenevä otsikointi välittyy ohjelmistokuvauksen kautta arkkitehtuuriin ja sovelluksen toteutukseen, jossa toiminnallisuudet ja piirit on nimetty yhtenevillä otsikkotiedoilla.

Tässä vaiheessa hahmotellaan tukidokumentaation osalta käyttöohjetta sekä testaussuunnitelmaa integrointitestauksen ja järjestelmätestauksen kannalta. Lopuksi arkkitehtuurisuunnittelun, I/O:n kokonaismäärän sekä toiminnallisuuksien monimutkaisuuden perusteella toteutetaan suunnittelijan oman työn aikataulusuunnitelma, jonka toteutumista seuraataan. Ohjelmistokuvaus voidaan jättää tarkastamatta, jos lähtötietojen muuttumisuhka on merkittävä. Tyypillisiä tällaisia tilanteita ovat uusien ja vähän aikaa alalla olleiden toimijoiden kanssa toimittaessa.

Toteutusvaiheen laadunvarmistus tapahtuu ohjeellisen mallin mukaan testaamisella sekä kokeneemman suunnittelija suorittamalla läpikäynnillä (walkthrough) ja

järjestelmätestauksessa suoritetulla tarkastuksella. Jokaisen arkkitehtuurisuunnitelman mukaisen ohjelmalohkon toteutuksen jälkeen se testataan ohjelmistokuvausta vastaan. Tämä tapahtuu suunnittelijan itse suorittamalla moduulitestauksella, josta ei kirjoiteta testausraporttia, eihän testaussuunnitelmaakaan ollut tehty moduulitasolle. Uupuvat testiraportit korvataan kirjaamalla pöytäkirjaan ohjelmamoduulien suoriutuminen testeissä.

Koska tässä projektissa automaatiosovellus toteutetaan kyseiselle suunnittelijalle ennestään tuntemattomalla teknologiaratkaisulla, tulee sovelluksen oikeaoppisesta kehittämisestä varmistua heti alkumetreillä. Toimilaitepiirien toteuttamisessa käytettävä valvomon & logiikkaohjelman toimilaitepiirien generointityökalu käydään kokeneemman suunnittelijan kanssa läpi. Myös automaatiolaitteiston konfiguraatioon liittyvissä asetuksissa hyödynnetään läpikäyntiä. Integrointitestaus suoritetaan ainoastaan niille ohjelman moduuleille jotka toimivat vuorovaikutuksessa keskenään. Toimilaitepiirien sekä muiden uudelleen käytettävien jo valmiiden ohjelmamoduulien kanssa keskustelevia moduuleja ei integrointi testata, koska kyseessä on moneen kertaan aikaisemmin testatut komponentit. Kuitenkin nämäkin komponentit testataan myöhemmin järjestelmätestauksen aikana.

Kehitysehdotuksen mukaisella ja ohjeelliseen malliin päätyneellä järjestelmätestauksella projektiin nimetty pääsuunnittelija tarkastaa kokonaisuuden ennen tehdastestauksen alkua. Tämä tilaisuus pidetään luonteeltaan muodollisena läpikäyntinä, josta kirjataan pöytäkirja erityisesti havaittujen poikkeavuuksien osalta. Järjestelmätestauksessa todennetaan automaatiolaitteiston ja väylän toiminta sekä näihin liitetyiden käyttöliittymäkomponenttien toiminta. Toiminnallisuuksia kohdellaan läpikäynnin mukaisella tavalla. Sähkö-automaatiokeskus tarkastetaan tämän valmistuttua ja varmistetaan että riviliitinrajapintaan asti kaikki on johdotettu oikein. Tästä kirjataan myös I/O-tarkastuspöytäkirja.

Järjestelmätestauksessa ja varsinaisessa tehdastestauksessa valvomo-ohjelmistoa suoritetaan virtuaalikoneessa, koska varsinainen valvomolaitteisto johon käyttöliittymän operointikuvat lisätään on loppuasiakkaan prosessissa oleva palvelinkokonaisuus. Tehdastestauksen mukainen kelpoistustoiminto on määritelty tässä projektissa sopimuksessa. Tämä toteutetaan muutaman päivän kestävänä, sovelluksesta ja käyttöliittymästä muodostuvan järjestelmän testauksena jäädytettyjä lähtötietoja tai projektin perustamisen jälkeistä perussuunnittelua vasten.

Asennustöiden valmistuttua sopimuksen velvoittamana asiakas avustaa piirikoestuksessa, jossa testataan väylä- sekä I/O-kytketyt toimilaitteet. Tämän jälkeen järjestelmää voidaan alkaa koekäyttämään. Automaatiotoimittaja vastaa oman toimituksensa käyttöönotosta ja asiakkaan vastuulle jää prosessin koekäytöstä vastaaminen. Kun kylmätestaus ja kuumatestaus on suoritettu osana käyttöönottoa, suoritetaan asiaankuuluva koulutus. Ohjeellisessa mallissa osana käyttöönottoa olevista hy-

väksyntätestauksesta tai kelpoistusjaksosta ei tässä vaiheessa ole tietoa tämän projektin osalta. Käyttöönoton päätteeksi projektinhallinnallisena toimenpiteenä katselmoidaan loppudokumentaation sekä kaikkien vaadittavien dokumenttien ja sovellusten tila sekä tehdään suunnitelma näiden viimeistelemiseksi ja arkistoimiseksi osana teknistä katselmointia.

7.2.2 Asiantuntija-arvio kehitysehdotuksen soveltuvuudesta

Työn sisältöä ja soveltuvuutta kohdeyrityksen tarpeisiin arvioi Instan projekti-insinööri Joonas Eirola [8] seuraavasti:

Kokonaistoimitusyksikön automaatioprojektien toteutus noudattaa pääsääntöisesti samaa kaavaa ja lainalaisuuksia kuten diplomityössäkin on kuvattu. Eroavaisuuksia aiheuttavat etenkin projektin laajuus, loppuasiakas sekä projektissa mukana oleva henkilöstö. Yrityksen kilpailuvaltti on ollut kustannustehokas ja riittävän laadukas työskentely. Tämän on mahdollistaneet asiansa osaavat suunnittelijat.

Diplomityössä esitetty malli soveltuu hyvin kohdeyrityksen toimintaan ollen riittävän kevyt ja selkeä, jotta se olisi mahdollista käyttöönottaa osaksi normaalia suunnitteluprosessia. Diplomityö käy monilta osin myös osana uuden suunnittelijan perehdytystä yrityksen toimintatapoihin.

Ajoittain on ilmennyt puutteita toteutuksen laadussa. Vaikka puutteet huomataan ennen toimitusta asiakkaalle, aiheuttaa korjaavat toimenpiteet kustannustehokkuuden heikentymistä. Pahimmillaan laatupuutteet voivat aiheuttaa huonoa mainetta yritykselle. Lisäksi haasteita muodostuu tiedon siirtämisestä projektin päätyttyä. Suurissa usean suunnittelijan projekteissa dokumentoinnin ansiosta ei esiinny ongelmia, mutta tyypillisen kokoisissa projekteissa dokumentaatiota ei muodosteta samoissa määrin. Tällöin pienempien projektien jälkihoito vaikeutuu, joka on merkittävä osa liiketoimintaa kun rakennetaan pitkiä asiakassuhteita [8].

Diplomityössä kuvattu malli pyrkii kehittämään sovellussuunnitteluprosessia pienin panostuksin sulauttamalla laadunvarmistus ja dokumentointi osaksi vakioitua projektin toteutusmallia. Kouluttamalla projektihenkilöstöä, luomalla heille helppokäyttöisiä työkaluja ja ohjaamalla toimintaa haluttuun suuntaan voidaan toteutusmalli vakiinnuttaa osaksi normaalia projektitoimintaa. Dokumentaatio saatetaan kokea ylimääräiseksi rasitteeksi ja laadun tarkkailu ahdistavaksi työn vakoiluksi, mutta jos ne ovat osa normaalia toimintatapaa ja niitä toteutetaan projektin aikana normaalin suunnittelun ohessa, ei niihin kuluva aika ja vaiva muodostu suureksi. Päinvastoin ne helpottavat projektin luovutusta asiakkaalle sovitussa ajassa ja varmistavat, että jälkihoito on mahdollista [8].

Ohjelmistokuvaus on tärkeä dokumentti, koska sen avulla suunnittelija voi yhdessä asiakkaan kanssa varmistaa sovelluksen oikea toiminta. Se soveltuu hyvin myös testaussuunnitelmaksi sekä myöhemmin käyttöohjeen pohjaksi.

Sovellussuunnittelussa on varmistuttava, että ohjelman vaatimukset on ymmärretty oikein ja näin ollen toteutetaan oikeita asioita tekevä sovellus. Diplomityössä esitetty malli painottaa myös ohjelmakuvauksen merkitystä projektin lähtötietojen ymmärtämiseksi ja sovelluksen toteutuksen ja testauksen pohjaksi [8].

7.3 Laadunvarmistuksen prosessin kehittämisehdotus

Laadunvarmistusprosessin kehittämisen seurantaan vaaditaan mittaustoimenpide työprosessiin tuotujen muutosten seuraamiseen. Esitetyssä kehitysehdotuksessa laadunvarmistukseen esiteltiin otettavaksi mukaan tarkastuksien yhteydessä havaittujen virheiden kirjaaminen. Jotta tämä voidaan aloittaa, määritellään virhetyyppikategoriat joihin havaitut virheet lokeroidaan. Merkittävänä asiana työilmapiirin takia mainittiin aikaisemmin luvussa 4.2 työntekijöiden informoimisesta, mihin kerättyä virheinformaatiota tullaan käyttämään. Toisen työstä virheiden etsiminen aiheuttaa helposti närkästystä, joten henkilökunnan perehdyttäminen aiheeseen on kehityksen onnistumisen kannalta isossa asemassa. Alkuvaiheessa otetaan käyttöön projektikohtainen systemaattinen laadunvarmistus läpikäyntien muodossa, jossa erillisen laadunvarmistajan toimesta laadunvarmistuksen tapahtuma tulee suunnittelijoille aluksi tutuksi. Tässä suosituksessa ehdotetaan epämuodollisten läpikäyntien lisäksi otettavan käyttöön muodollisempia laadunvarmistustoimia. Tarkoituksena on havaittujen virheiden osalta datan kerääminen ja tallentaminen myöhempää analysointia varten.

Lyhyen aikavälin kehityksessä epämuodollisten tarkastusten lisäksi ryhdyttiin muodostamaan dokumenttipohjia suunnittelun tukimateriaalia ajatellen sekä tarkastuslistoja laadunvarmistamisen ohjaamiseksi. Suosituksessa ehdotetaan näiden jatkokehittämistä sekä projektityyppikohtaisten ohjeiden muodostamista, jotka kuvaavat mitä tukimateriaalia ja minkälaisia laadunvarmistustoimia suoritetaan erityyppisissä projekteissa. Projektityyppejä voi olla vaikea muodostaa kerralla sellaisiksi, että kaikki tulevat projektit sijoittuisivat johonkin tyyppiluokkaan. Siksi kaikki edellä mainitut asiat pidetään jatkuvan kehityksen alla, jolloin ajan saatossa saadaan muodostettua toimiva mutta mahdollisimman kevyt kokonaisuus oikeasti hyödyllisistä toimenpiteistä sekä tukimateriaalista.

Suosituksen mukaisesti mahdollisimman aikaisessa vaiheessa havaitaan virheitä tarkastettavasta materiaalista. Tarkastuskohteet kuvattiin tarkemmin ohjeellisessa toimintamallissa ja tarkastamiseen sopivia laadunvarmistusmenetelmiä esiteltiin kappaleessa 5.1 Laadunvarmistus. Näillä toimenpiteillä kehityksen etenemistä voi-

daan alkaa yhdessä jo olemassa olevien laatumittareiden yhteisvaikutuksen avulla. Virheiden etsimiseksi ja kirjaamiseksi suoritettava laadunvarmistaminen vaikuttaa jo itsessään sovellussuunnittelun ilmapiiriin. Kun suunnittelija tietää että hänen työlleen tullaan tekemään tarkastus, ei hän päästä käsistään mitä tahansa tarkastettavaksi. Suunnittelija siis suoriutuu työstään täysin toisella tavalla kuin aikaisemmin. Jo tämä itsessään auttaa kohentamaan sovellusten laatua oikeellisuuden osalta. Ulkopuolisen laadunvarmistuksen seurauksena myös suunnittelijalle tulee käsitys, minkälaisiin virheisiin hän on taipuvainen. Tätä kautta hän pystyy vaikuttamaan ennalta työnsä lopputulokseen ja kehittymään suunnittelijana.

7.3.1 Lyhyen aikavälin kehitys

Sovellussuunnittelun työprosessin kehittämisessä, kuten minkä tahansa toimintaprosessin kehittämisessä, korostetaan muutoksen saavuttamiseksi kehityksen jakamista pieniin kerrallaan käyttöönotettaviin askeliin. Tätä käsitellään rekursiivisesti luvussa 2.2. Siksi lyhyen aikavälin aikana tehtävät muutokset tulisi olla mahdollisimman pieniä ja kevyitä. Niiden tulee tukea seuraavia kehitysvaiheita sekä ne tulisi saada jäämään käyttöön. Lyhyestä aikaväliä käsiteltäessä alkuvaiheen asiaksi sopii myös projektin tukidokumentaation dokumenttipohjien muodostaminen ja kokoaminen automaatiosuunnittelun toimintajärjestelmään. Tämän avulla projektin luonne huomioiden voidaan valita tarvittavat dokumentit, joiden muodostaminen suoritetaan dokumenttipohjien mukaisesti. Dokumenttipohjat ovat myös helppo toteuttaa, koska näihin voidaan käyttää vanhoista projekteista peräisin olevia dokumentteja joista riisumalla saadaan tarvittavan kattava pohjarakenne.

Ensimmäinen asia joka kehitysehdotuksessa tulisi suorittaa, on sellaisen laadunvarmistusmekanismin systemaattinen hyödyntäminen jota jo nyt kohdeyrityksessä on tietyissä tapauksissa käytetty. Kyseessä on siis epämuodollisen tarkastamisen tai läpikäynnin käyttöönotto. Edelleen tämäkin tekijä pysyisi projektikohtaisesti suunniteltavana asiana. Tarkastuksen laajuus jäävät aina projektikohtaisesti päätettäviksi asioiksi, mutta tarkastuksen aihe ja suorittajan valitseminen kuitenkin otetaan mukaan jokaiseen projektisuunnitelmaan laajuudesta riippumatta. Ensimmäiset tarkastajat muodostavat tarkastuslistat yleiseen käyttöön, joita kehitetään kattavammiksi ja tehokkaammiksi tulevien tarkastusten yhteydessä.

Painotuksena tulee olla tarkastuslistojen joustava soveltuvuus erityyppisiin projekteihin sekä oleellisten asioiden painottaminen. Projektin laajuuteen koodin määrältään ja I/O-määrältään on kuitenkin hankala ottaa kantaa tarkastuslistoja muodostaessa näin aluksi. Kuitenkin tiivis rakenne selkiyttää tarkastusta ja mahdollistaa tarkastuksen suorittamisen listan ohjaamalla tavalla. Tässä ei kuitenkaan perehdytä tarkemmin tarkastuslistojen muodostamiseen, vaan jätetään tarkastuksen suorittajien tehtäväksi. Ohjelmistoteollisuuden alalla on tarjolla ohjelmiston laadunvarmis-

tukseen liittyviä tarkastuslistoja joista saa käsitystä listan laajuudesta ja käsiteltävistä asioista.

7.3.2 Pitkän aikavälin kehitys

Automaation sovellussuunnittelun ja siihen liittyvän laadunvarmistamisen kehittäminen tulisi pitkän aikavälin näkymissä toteuttaa koko projektinhallinnan alueelta. Ohjelmistoteollisuuden puolella paljon käytetty on kypsyyden kehittämiseen perustuva jatkuva kehitys. Tähän on tarjolla aikaisemmin esitellyt CMMI ja SPICE (luku 2.2 sekä 4.6). Näistä CMMI on laajennettu käsittämään ohjelmistoteollisuuden tarpeista laajalle alueelle teollisuutta, painottaen kuitenkin edelleen sovellussuunnitteluun liittyviin asioihin.

Pitkän aikavälin kypsyystasoon perustuvassa kehityksessä ensin määritellään yrityksen lähtö kypsyystaso. CMMI-malli kuvaa kehityksen suorittamista jota varten muodostetaan erillinen organisaatio tai kehitys suoritetaan muun työn ohessa. Kypsyysmalli tarjoaa myös vaihtoehtoisia lähestymistapoja, joista löytyy sopivat toimintamallit myös automaatiosuunnittelua harjoittavaan organisaatioon. Kypsyystasoa kohottamalla yritys pystyy vastaamaan kilpailijoiden asettamiin haasteisiin joihin tukea antaa pitkälle kehitetyt työprosessit. Kehittynyt ja kypsä prosessi ei kuitenkaan takaa kilpailuetua verrattuna päteviin suunnittelijoihin, mutta kypsällä ja systemaattisella prosessilla jota suorittaa pätevät suunnittelijat saa aikaan parempaa tulosta kuin ilman sitä toimivat kilpailijat.

8. YHTEENVETO JA PÄÄTELMÄT

Automaationsuunnittelijan työprosessi sovellussuunnittelussa kartoitettiin kohdeyrityksen asiantuntijahaastattelujen avulla. Kirjallisuuden ohjelmistoprojekti- ja elinkaarimalleja sekä laadunvarmistustoimia soveltamalla työprosessista muodostettiin ohjeellinen malli. Suunnittelutyötä käsittelevä aihealue muodostuu käyttöautomaation sovellussuunnittelusta. Turvallisuuteen liittyvät järjestelmät ja korkeamman tason automaatioratkaisut on jätetty mallin sovellusalueen ulkopuolelle.

Haastattelujen myötä dokumentoitua prosessia on järjestelty soveltumaan kirjallisuudesta tunnistettuihin sovellussuunnittelun päävaiheisiin. Suunnitteluprosessin haasteisiin vastaamiseksi tukidokumentaation merkitys on nostettu esiin. Dokumentaation kehittäminen havaittiin oleelliseksi osaksi muita suunnitteluprosessien kehityshankkeita.

Kirjallisuusosiossa tunnistettuja elinkaarimalleja sekä laadunvarmistustoimia voidaan soveltuvin osin hyödyntää käyttöautomaation sovellussuunnittelussa. Inkrementaalisten mallien soveltuvuudessa havaittiin haasteita automaatioprojektien luonteesta johtuen, eikä ketterien elinkaarimallien joustavaa suhtautumista projektin aikaisiin vaatimusmuutoksiin voida tässä hyödyntää. Elinkaarimallien arvioinnissa soveltuvimmiksi nousevat perinteiset vesiputousmalli ja v-malli.

Muodostetussa ohjeellisessa mallissa (OASP) ehdotetaan hyödynnettäväksi tukidokumentaatiosta ohjelmistokuvausta. Ylläpitämällä dokumenttia koko suunnitteluprojektin ajan, tukee tämä kehityksen aikaista resursointia, toimii se porttina muutoksille, sekä ohjaa sovelluksen ylläpitoa. Varsinaiseen muutostenhallintaprosessiin ei tässä työssä ole lähdetty esittämään kehitystä, joten kirjallisuusosiossa esitetään ainoastaan muutostenhallinnan rakennetta ylemmältä tasolta.

Työlle asetetut tavoitteet saavutettiin sovellussuunnittelun työprosessin dokumentoinnin osalta sekä laadunvarmistuksen prosessin kehittämisen osalta muodostetulla kehitysehdotuksella. Edellä mainituista muodostettiin tässä esiteltävä ohjeellinen malli yrityksen toimintajärjestelmään. Malli voitiin toteuttaa sellaisilla käsitteillä ja niin yleiseltä tasolta, että se ei ota kantaa toimialaan eikä projektin laajuuteen. Mallin sisältämissä tehtävissä mukana oleva ehdollinen valinnan vapaus mahdollistaa liikkumavaraa projektien vaihtelevan luonteen mukaan. Tämän avulla täyttyy vaatimus mahdollisimman kevyestä laadunvarmistuksen prosessista, joka ei aseta toimintajärjestelmän kautta liian raskaita byrokratiavaatimuksia.

Turvallisuuskriittisten järjestelmien käyttöautomaation sovellussuunnittelua muo-

dostettu malli käsittelee ainoastaan siinä esitettävien laadun todisteiden muodostamisen osalta. Johtuen toimialakohtaisten turvallisuuskriittisten standardien valtavasta kirjosta malli ei kuitenkaan ota kantaa mitä tukidokumentaatiota ja mitä laatutodisteita näissä tulisi muodostaa.

Esitetyt laadunvarmistuksen toimenpiteet vastaavat hyvin yrityksen kohtaamiin automaation sovellussuunnittelun haasteisiin. Lähtökohtaisesti suunnitteludokumentaation tarkastaminen ennen suoritettavaa toteutusta mahdollistaa virheiden havaitsemisen aikaisessa vaiheessa. Toteutusvaiheen läpikäynnillä varmistetaan suunnittelijan oikeat käytänteet. Tilaisuus toimii myös koulutustapahtumana sekä tietotaidon yhtenäistämistapahtumana.

Tehdastestauksessa esiintyvät ongelmat eivät yleisesti anna hyvää kuvaa yrityksestä asiakkaille, jotka tutustuvat ensimmäistä kertaa automaatiototeutukseen käyttöliittymän välityksellä. Sijoittamalla tarkastajan toimesta toteutettava järjestelmätestaus suoritusajankohdaltaan ennen tehdastestausta, voidaan huolehtia asiakkaalle muodostuvan hyvä kuva järjestelmän toteutuksesta. Lisäksi laadunvarmistustilaisuuksista ryhdytään kirjaamaan havaittuja virheitä, joiden avulla saadaan yksi laatumittari lisää organisaation toiminnan seurantaan. Toteutuneiden aikataulujen, asiakastyytyväisyyskyselyn sekä nyt lisätyn mittauksen muodostamien tilastojen avulla kehityksen seuranta tehostuu. Näin suunnittelija tulee myös itsetietoisemmaksi virhetyypeistään joihin hän on taipuvainen.

Tukidokumentaation muodostaminen ja tarkastusten suorittaminen tulee vaatimaan työpanosta, joten nämä tulee huomioida resursoinnissa. Käytettävä aika suuntautuu kuitenkin tasapuolisesti suunnittelijan sekä tarkastajana toimivan henkilön kesken. Suunnittelija panostaa tarkastettavaan materiaaliin enemmän kun tietää, että sille tehdään julkinen tarkastus. Tällöin valmiimman ja virheettömämmän materiaalin tarkastaminen vähentää tarkastamiseen kuluvaa aikaa, joka puolestaan vähentää tarkastuksista tulevaa kuormaa sitä suorittaville eksperteille. Lisäksi tarkastettu ja dokumentoitu materiaali maksaa siihen käytetyn ajan takaisin moninkertaisesti tilanteissa, joissa projektin loppupuolella etsitään ja korjataan virheitä.

Käyttöönoton jälkeen sijoitettavaan tekniseen katselmointiin mennessä kaikki tukidokumentit ja muutokset ovat päivitetty ja valmiit. Tämän myötä päättyvät projektit ovat järjestelmällisesti viimeistelty ja projektit saadaan päättymään täsmällisesti. Kokonaisuudessaan tässä työssä esitettävä muutokset ovat pieniä, mutta niiden käytäntöön laittaminen on muutosten luonteesta johtuen suunnittelua ja työtä vaativaa, joten muutosten jalkauttaminen tapahtuu työn ulkopuolella.

LÄHTEET

[1] Ajo, R. (2001). Laatu automaatiossa: parhaat käytännöt. Suomen automaatioseura. Saatavissa: http://www.automaatioseura.com/automaatio/tiedostot/finish/17/188

[2] Asmala, H. (2005). Automaatiosovellusten ohjelmistokehitys: suunnittelun työtavat, välineet ja sovellusarkkitehtuurit. Suomen automaatioseura.

[3] Boehm, B. (1986). A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes, 11(4), s. 14-24.

[4] Canet, G., Couffin, S., Lesage, J. J., Petit, A., & Schnoebelen, P. (2000). Towards the automatic verification of PLC programs written in Instruction List. In Systems, Man, and Cybernetics, 2000 IEEE International Conference, 4, s. 2449-2454. IEEE.

[5] Cleland-Huang, J., Gotel, O., & Zisman, A. (2012). Software and systems traceability 2(3), s. 9. Springer.

[6] Douglass, B. P. (1998). Real-Time Design Patterns. Real-Time UML: Developing Efficient Objects for Embedded Systems. Addison-Wesley.

[7] Ehsan, N., Perwaiz, A., Arif, J., Mirza, E., & Ishaque, A. (2010). CMMI/SPICE based process improvement. In Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference, s. 859-862. IEEE.

[8] Eirola, J. (2016, huhtikuu) Projekti-insinööri. Asiantuntija-arvio.

[9] Fagan, M. E. (1974). Design and code inspections and process control in the development of programs. International Business Machines Corporation, System Development Division.

[10] Fisher, M. S. (2007). Software verification and validation: an engineering and scientific approach. Springer Science & Business Media.

[11] Haikala, I., Mikkonen, T. (2011). Ohjelmistotuotannon käytännöt. 12., uudistettu painos, Kariston kirjapaino Oy, Hämeenlinna. Talentum Media Oy.

[12] Haikala, I., Märijärvi, J. (2004). Ohjelmistotuotanto. 10. painos, Helsinki. Talentum.

[13] Hanhela, A. (2015, huhtikuu 30). Ryhmäpäällikkö, kehitys ja erikoisprojektit. Haastattelu.

[14] Hanhijärvi, J. (1991). Vaaranalainen ohjelmoitava automaatio kansainvälisten ohjeiden, standardien ja määräysten valossa. Esiselvitys. Valtion teknillinen tutkimuskeskus. Espoo.

[15] Heikkinen, H. L., Rovio, E., and Syrjälä, L. (2007). Toiminnasta tietoon: toimintatutkimuksen menetelmät ja lähestymistavat, Helsinki.

[16] Huuha, J. (2015, syyskuu 3). Projekti-insinööri, kehitys ja erikoisprojektit. Haastattelu.

[17] IEEE Std 1028-2008. IEEE Standard for Software Reviews and Audits, Software & Systems Engineering Standards Committee of the IEEE Computer Society.

[18] Jetley, R., Nair, A., Chandrasekaran, P., & DUBEY, A. (2013). Applying software engineering practices for development of industrial automation applications. In Industrial Informatics (INDIN), 2013 11th IEEE International Conference, s. 558-563. IEEE.

[19] Järvenpää, S. (2015, toukokuu 22). Ryhmäpäällikkö, Ympäristö- ja vesihuolto. Haastattelu.

[20] Komu, T. (2015, kesäkuu 24). Ryhmäpäällikkö, elintarvike- ja kemianteollisuus. Haastattelu.

[21] Kuikka, S. (2013). Automaation reaaliaikajärjestelmät, automaaatio-ohjelmistojen laadusta. Opetusmateriaali [PowerPoint -kalvot]. Tampereen teknillinen yliopisto. Systeemitekniikan laitos.

[22] Leppälä, H. (2003). Laadunvarmistusprosessin kehittäminen kaupankäyntiohjelmistojen kehityshankkeissa. Diplomityö. Tampereen teknillinen yliopisto. Tietotekniikan koulutusohjelma.

[23] Leppänen, J-P. (2015, toukokuu 18). Projekti-insinööri, ympäristö- ja vesihuolto. Haastattelu.

[24] Mahlanen, A. (2015,lokakuu 9). Projekti-insinööri, energia- ja prosessiteollisuus. Haastattelu.

[25] Mäkinen, P. (2002). Reflektio oppimisessa. Saatavissa: `http://www15.uta.fi/arkisto/verkkotutor/reflekt.htm`

[26] Mäkäräinen, M. (2000). Software change management process in the development of embedded software. Väitöskirja. Oulun yliopisto. Technical Research Centre of Finland. Espoo.

[27] Nevalainen, R., Harju, H., Halminen, J., & Johansson, M. (2010). Certification of software in safety-critical I&C systems of nuclear power plants. INTECH Open Access Publisher.

[28] O'Regan, G. (2010). Introduction to software process improvement. Springer Science & Business Media.

[29] Parnas, D. L., & Lawford, M. (2003). The role of inspection in software quality assurance. Software Engineering, IEEE Transactions on, 29(8), s. 674-676.

[30] Peik, A. (2002). Muutoksenhallinta tietojärjestelmän ylläpidossa. Diplomityö. Tampereen teknillinen yliopisto. Tietotekniikan koulutusohjelma.

[31] Pesonen, E. (2009). Ohjelmistoprojekti. Opetusmateriaali [web-sivu]. Itä-Suomen yliopisto. Tietojenkäsittelytieteen koulutusohjelma. Saatavilla: `http://www.cs.uku.fi/kurssit/opr`

[32] Pressman, R. (2001). Software Engineering: A Practitioner's Approach. 5. uudistettu painos. McGraw-Hill. New York.

[33] Rook, P. (1986). Controlling software projects. Software Engineering Journal, 1(1), s. 7.

[34] Royce, W. W. (1970). Managing the development of large software systems. In proceedings of IEEE WESCON, 26(8), s. 328-388.

[35] Savolainen, K. (2015, heinäkuu 13). Projekti-insinööri, elinkaari- ja ylläpitopalvelut. Haastattelu.

[36] SFS-EN 61508-3. (2011). Sähköisten/elektronisten/ohjelmoitavien elektronisten turvallisuuteen liittyvien järjestelmien toiminnallinen turvallisuus. Osa 3: Ohjelmistovaatimukset. Suomen standardisoimisliitto.

[37] SFS-EN ISO 9001:2008. Laadunhallintajärjestelmät. Vaatimukset. Suomen standardisoimisliitto.

[38] SFS-IEC 61506. (1998). Teollisuusprosessien mittaus ja ohjaus. Sovellusohjelmiston dokumentaatio. Suomen standardisoimisliitto.

[39] Sommerville, I. (2011). Software Engineering. Pearson, cop. Boston.

[40] Stephens, M., & Rosenberg, D. (2007). Use Case Driven Object Modeling with UML: Theory and Practice.

[41] Systä, K., Leppänen, M. (2013). Ohjelmistotuotannon menetelmät. Opetusmateriaali [PowerPoint -kalvot]. Tampereen teknillinen yliopisto. Ohjelmistotekniikan laitos. Saatavilla: http://www.cs.tut.fi/~otm/luennot/kalvot/

[42] Tamminen, P. (1994). Laatujärjestelmän kehittäminen ohjelmistotuotannossa. Diplomityö. Tampereen teknillinen korkeakoulu. Tietotekniikan koulutusohjelma.

[43] Team, C. P. (2010). CMMI® for Development, Version 1.3, Improving processes for developing better products and services. no. CMU/SEI-2010-TR-033. Software Engineering Institute.

[44] Technical report series No. 384, Verification and Validation of Software Related to Nuclear Power Plant Instrumentation and Control (1999). International atomic energy agency, Vienna.

[45] Thramboulidis, K., Soliman, D., & Frey, G. (2011). Towards an automated verification process for industrial safety applications. In Automation Science and Engineering (CASE), 2011 IEEE Conference, s. 482-487. IEEE.

[46] Thramboulidis, K., & Frey, G. (2011). Towards a model-driven IEC 61131-based development process in industrial automation. Journal of Software Engineering and Applications, 4(04), s. 217.

[47] Wang, L., & Tan, K. C. (2006). Modern Industrial Automation Software Design. John Wiley & Sons.

[48] Wiegers, K. E. (2003). Software Requirements: Practical techniques for gathering and managing requirement through the product development cycle. Microsoft Corporation.

[49] Vartiainen, A. (2012). Ohjelmistoprojektien laadulliset haasteet. Opinnäytetyö. Haaga-Helia. Tietojenkäsittelyn koulutusohjelma.

[50] Vuori, M. (2011). Agile development of safety-critical software. Tampereen teknillinen yliopisto. Ohjelmistotekniikan laitos. Raportti 14.

[51] Vuorinen, J. (2010) Scrum-menetelmän käyttö Pirkanmaalaisissa ohjelmistoyrityksissä. Diplomityö. Tampereen teknillinen yliopisto. Tietotekniikan koulutusohjelma.

[52] Ylinen, A. (2015, joulukuu 11). Projekti-insinööri, elinkaari- ja ylläpitopalvelut. Haastattelu.

LIITE A:
AUTOMAATIOSOVELLUKSEN TARKASTUSLISTA

Projekti	
Projektipäällikkö	
Suunnittelija	
Tarkastaja	
Tarkastuspäivämäärä	

Tukidokumentit, nimi ja versio *(I/O-luettelo, ohjelmistokuvaus, käyttöohje)*

PLC kehitysohjelmiston versio	
Valvomon kehitysohjelmiston versio	
Paneelin kehitysohjelmiston versio	
Ylemmän tason automaatiosovellusten versiot	

Tarkastuksen kohde	OK	poikkeus	NA	Huom
HW-konfiguraatio				
PLC käännösvirheet				
Toimilaitepiirit				
Rajapinnat				
Tukidokumenttien tila				
Koodin ulkoasu				
Koodin kommentointi				
Käyttöliittymän virheet				
Järjestelmätestausmenettelyt				
Ylemmän tason automaatiosovellusten konfigurointi				
Toiminnallisuuksien pistokokeet:				

Tarkastuslistan poikkeukset Jos jokin tarkastuslistan itemeistä poikkesi odotetusta, tähän selite (mikä mitä miksi)

LIITE B:
OHJELMISTOKUVAUS DOKUMENTTIPOHJA

SISÄLLYSLUETTELO

1 JOHDANTO 1

2 ARKKITEHTUURI 1

3 PROSESSI 1

3.1 ÖLJYKIERTO 2

3.1.1 Jäähdytysöljy 2

3.1.1.1 Positiot 2

3.1.1.2 Asetukset 2

3.1.1.3 Kuvaus 3

3.2 OSAPROSESSI 1

3.2.1 Toiminnallisuus 1

3.2.1.1 Positiot 1

3.2.1.2 Asetukset 1

3.2.1.3 Kuvaus 2

4 PIIRIT 4

4.1 MITTAUKSET 4

4.1.1 1-LE1 Öljynpintamittaus 4

1 JOHDANTO

Johdantoon lyhyt esittely projektista. Johdannon tarkoituksena on perehdyttää dokumentin lukija automatisoitavaan kohteeseen. Käsiteltävinä asioina automatisoitava kohdeprosessi ja automaatiolaitteisto sekä kytkeytyminen muihin järjestelmiin. Tässä mallidokumentissa on ohjeistavat tekstit kursivoitu, hypoteettiset esimerkit on eritelty oranssilla.

Mallidokumentti pohjautuu yrityksessä toimintakuvausten muodostamiseen perustuvaan dokumentointikäytänteeseen, joka sovellussuunnittelijan toimesta toteutettuna vastaa pienin lisäyksin kattavaa ohjelmistokuvausesitystä.

2 ARKKITEHTUURI

Tässä kuvataan tämän ohjelmistokuvauksen perusteella osakokonaisuuksiin jaotellun automaatiosovelluksen rakenne.

3 PROSESSI

Ohjelmitokuvaukseen kootaan otsikoiden alle lähtötietojen perusteella yksi kokonaisuus tietyn toiminnallisuuden toteuttamiseksi. Luvussa esitellään ensin kokonaisuuteen liittyvät automatisoidut laitteet positioineen, joiden kanssa kuvatussa kokonaisuudessa vuorovaikutetaan. Seuraava pääluku on tarkoitettu näiden esittämiseksi piirikohtaisesti ja tarkemmin.

Laitteiden kuvauksien jälkeen esitellään vaadittavat asetusarvot joiden avulla ohjelmalohko parametroidaan käyttöliittymästä sekä indikoinnit joilla esitetaan tietoja käyttöliittymään. Kyseessä on toimilaitepiirien ulkopuolisten kenttälaitteiden tilojen esittämistä sekä ohjelmalohkon sisäisien tilojen esittämiseen. Näiden jälkeen luvun lopuksi muodostetaan toimintakuvauksesta yksiselitteinen esitys toteutettavista toiminnallisuuksista.

3.1 OSAPROSESSI

3.1.1 TOIMINNALLISUUS

3.1.1.1 POSITIOT

Koneet

Binääritulot

3.1.1.2 ASETUKSET

Valinnat
val_1

Asetusarvot
as_1

Viiveet
t_1

Arvot

ar_1

3.1.1.3 KUVAUS

3.1.1.3.1 Toiminnallisuus 1

3.1.1.3.2 Toiminnallisuus 2

3.2 ÖLJYKIERTO

3.2.1 JÄÄHDYTYSÖLJY

3.2.1.1 POSITIOT

Koneet

Positiotunnus	*Prosessilaitteen nimi (prosessilaitteen positio)*
P-101	Jäähdytysöljypumppu (1-L001)

Venttiilit

Positiotunnus	*Venttiilin nimi prosessin perusteella (prosessilaitteen positio)*
1-SV1	Jäähdytysöljyn säätöventtiili

Mittaukset

Positiotunnus	*Mitattavaa suuretta kuvaava nimi*
1-LE1	Jäähdytysöljysäiliön pinnanmittaus

Binääritulot

Positiotunnus	*Tuloindikaation nimi (prosessilaitteen positio)*
1-LS-1	Jäähdytysöljyn ylärajan pintakytkin

PID-säätimet

Positiotunnus	*Säädettävän suureen nimi*
1-PIC-101	Jäähdytysöljy, paineensäädin

3.2.1.2 ASETUKSET

Valinnat

val_1	*Ohjaustavan kuvaus, parametrien merkitys ja syötteet [bool, integer]*
val_2	Puhallin ½ teho: Päällä / Pois
val_3	Ajotapa | 1 = pintasäätö, 2 = käynnistysraja-pysäytysraja, 3 = pintakytkin (LS-1)

Asetusarvot

as_1	*Käyttöliittymästä annettava parametri ohjelmalohkolle (insinööriyksikkö)*
as_2	Pinnankorkeuden asetusarvo (m)
as_3	Käynnistysraja (m)
as_4	Pysäytysraja (m)

Viiveet

t_1	*Kuvaus (insinööriyksikkö)*
t_2	Ajotapa: Pintakytkin | Käynnistysviive (s)
t_3	Ajotapa: Pintakytkin | Pysäytysviive (s)

Arvot

ar_1	*Kuvaus (insinööriyksikkö)*
ar_2	Ajotapa: Pintakytkin | Käynnistysviiveen kulunut aika (s)
ar_3	Ajotapa: Pintakytkin | Pysäytysviiveen kulunut aika (s)

Indikoinnit

id_1	*Kuvaus*
id_2	Jäähdytysöljyn alarajan pintakytkin
id_3	Huoltohälytys: jäähdytysöljyn paine

3.2.1.3 KUVAUS

3.2.1.3.1 Ohjaustapa: Pintasäätö/Käynnistys-pysäytys rajat/Pintakytkin

Kuvaus sovelluksen toteuttamista toiminnallisuuksista. Kielenä esimerkiksi yksiselitteinen luonnollinen kieli. Jos kyseessä on monimutkaisempi algoritmi, tarjoaa pseudokoodi tehokkaamman kuvaustavan.

Pintasäätö
Ajotapa (val_3) valittu pintasäädölle = 1.
Öljysäiliön (1-L001) pinta pidetään asetusarvossaan (as_2) säätimen 1-PIC-101 avulla ohjaamalla säätöventtiiliä 1-SV1.

Käynnistys-pysäytys rajat
Ajotapa (val_3) valittu käynnistys-pysäytys rajoille = 2.
Pinnanmittauksen (1-LE1) ylittäessä käynnistysrajan (as_3) ohjataan öljypumppu (P-101) käyntiin viiveen (t_2) jälkeen.
Pinnanmittauksen (1-LE1) alittaessa pysäytysrajan (as_4) pysäytetään öljypumppu (P-101) viiveen (t_3) jälkeen.

Pintakytkin
Ajotapa (val_3) valittu pintakytkimelle = 3.
Pintakytkimen ollessa aktiivisena viiveen (t_2) ajan käynnistetään öljypumppu (P-101) viiveen (t_3) ajaksi.

3.2.1.3.2 Öljynpaineen tarkkailu

…

3.2.1.3.3 Öljysäiliön tyhjennys

…

4 PIIRIT

4.1 MITTAUKSET

4.1.1 1-LE1 ÖLJYNPINTAMITTAUS

Mallipiiri	Mittaus
Mallipiirin kuvaus	Mittaus
Mittausalue	0..5
Mittausyks.	m
Keskus	A1.CPU

Rajat
Ylärajahälytys
Ylärajavaroitus
Alarajavaroitus
Alarajahälytys

Toiminta
3.1.1 Jäähdytysöljy

Lähtevät lukitukset

Raportointi
Keskiarvo
Minimi
Maksimi

Huom.